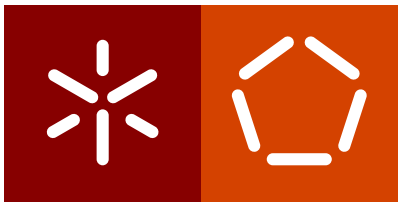
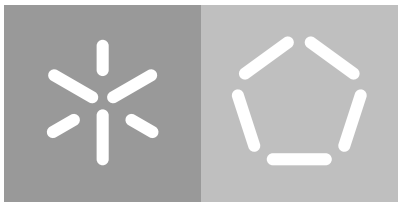


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Nuno Miguel Teixeira dos Santos

A feasibility study on the use of smartphone sensors for development of Advanced Driver Assistance Systems

June 2017



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Nuno Miguel Teixeira dos Santos

A feasibility study on the use of smartphone sensors for development of Advanced Driver Assistance Systems

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
João Miguel Fernandes
Helena Rodrigues

June 2017

To the late Ângelo Vilela, for persuading me to apply for a master's degree.
You will always be fondly remembered.
Obrigado.

ACKNOWLEDGEMENTS

This incredible journey would not be feasible without the help of many people. There are too much of them to fit on a single page, but an acknowledgment is granted to everyone who contributed to my education – in this particular case, especially to all the authors present in the reference list.

I'd also like to thank Prof. João Miguel Fernandes and Prof. Helena Rodrigues, for their guidance trough the whole dissertation process.

A very humble thanks to André Ferreira, for providing the opportunity to work with him at Bosch Car Multimedia and for supplying with all the tools needed for the experiments (sorry for any damage to the car). Also, for generously proofreading this document and for providing much valuable feedback and insights. I hope to continue learning from you as much as in these past months.

My profound gratitude goes to every member of my talented team. To Afonso Vilaça, for providing solid comments on physics- and statistics-related matters. To Eduardo Pinto, for designing and implementing the cloud application to where data was sent. To Nuno Dionísio, for assembling and maintaining a reliable infrastructure. To Pedro Cunha, for unknowingly being my rubber duck when debugging code. And to Rodrigo Carvalho, for all the priceless nuggets of knowledge provided every time we exchanged ideas while walking to the canteen.

To family and friends goes my warm recognition, for putting up with plenty of days with moodiness induced by sleep deprivation. Yes, parents, this means I will be around more frequently at sunday lunches. Yes, friends, this means I will show up more regularly at weekend dinners and parties.

A very special, heartfelt thanks to my girlfriend Diana Martins, for taking care of the house while I went to the mountains to write this. I promise to never let the dishes pile up in the sink again – yes, we're probably getting a dishwasher.

ABSTRACT

Technological evolution is impacting several industries, e.g., by allowing them to deliver higher levels of functionality. The automotive industry is an example of how technology is supporting the development of new solutions in vehicle safety and comfort.

Advanced Driver Assistance Systems (ADAS) are cases of solutions that evolved significantly in recent years. This is possible not only due to the progress of electronic solutions but also because of higher quality in software. The smartphone is an example of this evolution with a broad range of applicability since these devices have been used to develop ADAS, making them an interesting cost-effective platform to develop such systems.

Previous research has shown smartphones' ability to output sensors data with the necessary quality for a broad number of applications with special focus in inertial sensors. However, such studies tend to be difficult to reproduce or lack the desired detail levels of their experimental methods. Concerns about how good are smartphone sensors and their use to develop ADAS emerge when reading existing literature, particularly, how the context of collecting data is controlled and which variables impact the collection process.

In order to assess the feasibility of using smartphones as sensing devices, questions arise on how different parts of the collection setup affect the quality of data collected. Motivated by those questions, a study considering four different hypotheses is proposed to assess the impact of a controlled set of variables, namely: brands of inertial sensors, car mounts, sensor sampling rates, and vehicles. A set of controlled experiments is performed to assess the impact of each variable in the collection process of inertial sensors, more precisely the vertical acceleration.

To perform the experiments, three special-purpose tools were developed. Smartphones used in the experiments feature an application to collect and export their sensors data. A researcher of an experiment operates another smartphone application to annotate road anomalies found while driving. A desktop application automates the computation and statistical validation of the vertical acceleration correlation from different setups.

Dynamic Time Warping was used to compute the correlation coefficient of vertical acceleration as measured by different devices. Results show a baseline correlation coefficient of 0.892 with a standard configuration of software and hardware. When one of the independent variables is changed, the resulting coefficients range from 0.827 to 0.848.

Randomization tests were executed to statistically validate experiments results, making use of a Random Shuffle algorithm on surrogate data. Such tests rejected all four proposed null hypotheses regarding dissimilarities on vertical acceleration sensed by different setups.

From the controlled experiment a deeper understanding of the variables influencing data collection with smartphones was obtained. Results showed that varying the inertial sensors, car mounts, rates of sampling, or vehicles had a low impact on vertical acceleration sensed by smartphones. This is a good indicator that smartphones can be used to develop ADAS without the need to standardize every part of the collection setup. Thus, it possible to foresee the deployment of a system to a wider audience by taking advantage of existing equipment.

KEYWORDS Advanced Driver Assistance Systems, smartphones, inertial sensors, vertical acceleration, correlation coefficient, Dynamic Time Warping

RESUMO

A evolução tecnológica está a afectar várias indústrias, por exemplo, ao capacitá-las para fornecer níveis mais elevados de funcionalidade. A indústria automóvel é um exemplo da forma como a tecnologia está a apoiar o desenvolvimento de novas soluções de conforto e segurança automóvel.

Os Sistemas Avançados de Assistência ao Condutor – Advanced Driver Assistance Systems (ADAS) – são casos de soluções que evoluíram significativamente nos últimos anos. Para tal, não só contribuiu o progresso de soluções electrónicas, mas também o aumento de qualidade do software. Os smartphones são um exemplo desta evolução de ampla aplicabilidade, sendo já utilizados para desenvolver ADAS e uma interessante plataforma para desenvolver tais sistemas com baixo custo.

Estudos anteriores demostraram a capacidade dos smartphones para fornecer dados de sensores com a qualidade necessária para um grande número de aplicações, com especial foco nos sensores inerciais. No entanto, tais estudos tendem a ser de difícil reprodução ou não possuem o nível de detalhe desejado nos seus métodos experimentais. Questões sobre a qualidade dos sensores dos smartphones e o seu uso para desenvolver ADAS surgem do estudo da literatura existente, particularmente como a recolha de dados pode ser controlada e que variáveis têm impacto nesse processo.

Para avaliar a viabilidade do uso de smartphones como dispositivos sensoriais, nascem questões sobre como as diferentes partes do sistema afetam a qualidade dos dados recolhidos por ele. Motivado por essas questões, é proposto o estudo de quatro hipóteses para medir o impacto de um conjunto de variáveis, a saber: sensores inerciais, suportes de telemóvel, taxas de amostragem dos sensores, e veículos. Experiências controladas são realizadas para estudar o impacto de cada variável no processo de recolha de dados de sensores, mais precisamente a aceleração vertical.

Foram desenvolvidas três ferramentas de software para a realização das experiências. Os smartphones usados possuem uma aplicação para recolher e exportar os dados dos seus sensores. Durante a experiência, um investigador utiliza outra aplicação de smartphone para anotar as anomalias da estrada encontradas durante a condução. Uma aplicação de desktop automatiza a computação e validação estatistica da correlação da aceleração vertical medida por diferentes dispositivos.

O coeficiente de correlação da aceleração vertical medida por diferentes dispositivos fez-se usando o algoritmo Dynamic Time Warping. Os resultados mostram um coeficiente de 0.892 com uma configuração padrão de software e hardware, que serve como base de análise. Quando uma das variáveis independentes é alterada, os coeficientes resultantes variam entre 0.827 e 0.848.

Testes de permutação foram executados para validar estatisticamente os resultados experimentais, usando o algoritmo Random Shuffle sobre dados substitutos. Esses testes rejeitaram as quatro hipóteses nulas relativas à diferença de aceleração vertical detetada por diferentes dispositivos.

A partir das experiências obteve-se uma compreensão aprofundada das variáveis que influenciam a coleção de dados com smartphones. Os resultados mostram que variar os sensores inerciais, suportes de telemóvel, taxas de amostragem, e veículos tem baixo impacto na aceleração vertical detetada. Isto indica que estes dispositivos podem ser usados para desenvolver ADAS sem a necessidade de padronizar cada peça da recolha de dados. Assim, é possível antever o desenvolvimento de um sistema para um público mais amplo, tirando partido de equipamentos já existentes.

PALAVRAS-CHAVE Sistemas Avançados de Assistência ao Condutor (Advanced Driver Assistance Systems), smartphones, sensores inerciais, aceleração vertical, coeficiente de correlação, Dynamic Time Warping

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

LIST OF TERMS AND ABBREVIATIONS

ABS	anti-lock braking system. 4
ADAS	Advanced Driver Assistance System. iii, iv, 1, 2, 4–6, 13, 49, 50, 54
API	application programming interface. vii, 6, 16, 26, 27, 29, 30
CRUD	create, read, update, and delete. 30
CSV	comma-separated values. 27, 30
DTW	Dynamic Time Warping. iii, iv, 10, 11, 23, 24, 30, 45, 46, 50, 74, 77
ESP	electronic stability program. 4
GC	garbage collection. 26
GPS	Global Positioning System. 5–9, 20, 26, 33, 35, 38
HTTPS	Hyper Text Transfer Protocol Secure. 29
IDE	integrated development environment. 69
IMU	inertial measurement unit. 16, 18, 23, 51
MVC	Model–View–Controller. 26
MVP	Model–View–Presenter. 26, 30
OBD-II	on-board diagnostics. 8
OS	operating system. 6, 18, 34
P^2	Pothole Patrol. 6, 19
PC	personal computer. 7

REST representational state transfer. 29
RS Random Shuffle. iii, iv, 24, 31, 46
RTOS real-time operating system. 9, 23, 34

SNR signal-to-noise ratio. 7, 13, 23, 42, 44

UI user interface. vii, 24–27, 30

VIO Vehicles in Operation. 17, 50, 51

WLAN Wireless Local Area Network. 5

XDK Cross Domain Development Kit. 37, 55

1

INTRODUCTION

Both vehicles and smartphones are growing as sensing platforms with known cases of Advanced Driver Assistance System (ADAS) supported by the latter group. An understanding of the variables impacting the sensing capabilities will benefit the development and prototyping of ADAS.

1.1 CONTEXTUALIZATION

The development of services for the automotive sector has been progressing based on the technological evolution evidenced in recent years, namely the introduction of software in contexts where electromechanical solutions prevailed. All ADAS are examples of such advancements, like automatic parking or lane departure warning system, resulting from the combination of sensory capabilities and software in vehicles.

In parallel with the vehicle's ability to sense the surrounding environment, smartphones are increasingly endowed with relevant sensory capabilities, with existing cases of smartphone usage in the development of ADAS. The study described in this dissertation is justified by the uncertainties about the quality of data acquired from such devices for the development of ADAS.

The work described in this thesis aims to study the use of smartphone sensors for prototyping and developing ADAS. The expected outcome is a better understanding of the impact of the studied variables in the sensing capability of smartphones when used to develop ADAS.

1.2 MOTIVATION

A considerable number of ADAS rely on inertial data as the basis for their functionalities and there are inertial sensors embedded in the majority of smartphones available today. From these two premises, it follows that smartphones should be an interesting data aquisition platform for prototyping and developing ADAS.

Being fairly easy to obtain inertial data from these mobile devices, it is now required to study which variables impact the quality of data collected. The outcome of studying these

variables can influence the design and development of ADAS, determining if data can be reliable – or not – for the development of such systems and which considerations should be taken depending on the variables influencing – or not – the sensing capabilities for the desired purpose.

1.3 OBJECTIVE

This dissertation's main objective is to investigate the feasibility of using smartphone sensors to prototype and develop ADAS. Factors involved in the collection of sensors data will be studied to analyze the impact of such factors in the collection process.

This can happen by better understanding which variables influence the quality of data retrieved by inertial sensors embedded in smartphones. To ensure that this dissertation can be successfully completed in a reasonable amount of time, the scope of the studies sensors will be narrowed to the accelerometer and, from the collected data, only the vertical acceleration will be analyzed.

1.4 DOCUMENT STRUCTURE

The structure of this dissertation is based on the Experimental Software Engineering Process, proposed by Goulão and Abreu [1]. Their model aims to guide the planning, execution, and documentation of experiments in a way that promotes their reproducibility.[1] It is critical to construct an experiment by focusing on maximizing its ease of replication, in order to allow further research to either disprove its results or present more evidence of its validity.

This need arises from a growing concern within the scientific community on what has been described as a "replication crisis" [2]. Due to the extreme importance of this matter for the scientific method, authors have recently proposed that no paper should be published without an independent confirmation study being performed first [3].

Despite the fact that this problem affects more severely other disciplines, a 2016 poll from the journal Nature reported that $70\,\%$ of scientists in the engineering field have failed to reproduce at least one experiment from a peer. Furthermore, $50\,\%$ of the same scientists have failed to reproduce one of their own experiments [4].

This document comprises seven chapters with this introduction being the first. It describes the context of this dissertation, followed by the motivation for the problem subject of research, along with the definition of thesis objectives.

1 Despite being focused on experiments for the Software Engineering domain – e.g., studying the impact of new languages or tools –, the model proposed by Goulão and Abreu [1] is based on concepts described in empirical research guidelines [5, 6]. This signals that their approach has merits on contexts beyond the intended domain

In Chapter 2, the state of the art is presented. In addition to introducing multiple subjects to be handled in subsequent chapters, it presents a discussion on the merits of prior research from multiple authors.

Chapter 3 details the planning of the experiments. Decisions about the design of the experiment are described, alongside the justifications for those decisions. Hypotheses are formulated with dependent and independent variables selected for analysis. The experiment design is characterized and the collection process is defined. Then, analysis techniques are proposed and instrumentation is defined.

The execution of the experiments is reported in Chapter 4. For each experiment, a chronology of events if presented with additional details on configurations of hardware and software in use. Difficulties faced during the experiments are presented as well as solutions for such problems.

Chapter 5 contains the data analysis process. A statistical description of data collected is performed, followed by the detection and removal of incorrect values. It finishes by testing the hypotheses previously proposed by making use of techniques earlier defined.

Results are discussed in Chapter 6. In addition to the interpretation of results, it includes the identification of threats to the experiments' validity. Inferences are made on how the results are expected to hold for each variable's population and lessons learned are stated to aid researchers trying to replicate the experiments.

Finally, Chapter 7 includes the documentation of conclusions and proposals of future work.

Additionally, several auxiliary documents are compiled as appendices, including a study on the pervasiveness of sensors in smartphones, the requirements identified to build the data acquisition application for an Android smartphone used in the experiments, the implementation of a desktop application to automate the data analysis, and charts depicting the computed correlation coefficients.

2

STATE OF THE ART

This chapter will offer insights into the concepts addressed during the dissertation. After introducing the concept of ADAS, an overview is presented on the use of different sensors to feed them with different kinds of information.

Mobile devices, in particular smartphones, are discussed next. An exposition on the trend of smartphone sensors is made, followed by a discussion of the pros and cons of using smartphones for the development of ADAS. Then, the most prominent papers related to the collection of environmental information are discussed, along with their shortcomings.

Difficulties encountered during data analysis by previous researchers are described and frailties on their studies are explored. Such problems are usually related to data synchronization and the statistical methods used to analyze the data, so this is last section's focus.

2.1 ADVANCED DRIVER ASSISTANCE SYSTEMS

ADAS are electronic systems that aim to augment vehicles systems in order to improve road traffic safety, supporting the driver in their driving task. Lindgren and Chen [7] state that such support might range from simple information presentation, through advanced assisting, to taking over the driver's tasks in critical situations. A vehicle equipped with an ADAS is commonly referred to as an *intelligent vehicle* [8, 9] or as a *smart car* [10, 11].

Historically, research has been focused on passive protection systems, engineered to help the driver in the event of imminent crashes, like seat belts or airbags. Trying to overcome the obvious limitations on passive systems – since they are only useful when an accident occurs – investigation began shifting towards systems like the anti-lock braking system (ABS) [12] and the electronic stability program (ESP) [13].

Nowadays, according to Kim and Shin [14], the emphasis is changing from passive protection systems to active protection systems – from crash survival to crash avoidance, one could say. Such authors point out that the safety of the driver and of the people outside the vehicle is the objective of ADAS development, contrasting this approach with crash survival systems which only ensures the driver's safeness. An example of this active approach is the solution for pedestrian collision avoidance developed by Eckert et al. [15].

The long-term goal for ADAS is to provide a fully autonomous vehicle with self-driving capabilities and to guarantee an accident-free driving experience [14]. Shladover [16] expects automated highway systems with fully automated cars to significantly benefit traffic safety.

In order to reach the desirable accident-free driving experience, the autonomous vehicle should be capable of sensing its surroundings. This can be performed with a wide array of sensors embedded in the vehicle – with radar, camera, and ultrasound being the most common types currently used [17].

Kim and Shin [14] provide some insights as to why those three types are the commonplace: the low cost of ultrasonic sensors allows them to be easily added into a vehicle to provide proximity detection at low speeds; radars are pricier and bulkier but work from short to long ranges, at any speed, and any weather condition; and cameras, at a lower price, enhance other sensors by providing a 360º view and allowing the detection of objects.

A report from Texas Instruments [17] states that today's most ADAS functionalities exist in independent systems, noting a growing need to combine different sensor inputs in the future. According to the provided information, this will lead to more accurate decisions, higher system performance, and lower power consumption.

2.2 SMARTPHONES

Mobile devices, a category where smartphones and tablets are included, are now commonplace to nearly half of world's population [18]. They offer a great many of new capabilities and use cases, some of them being provided by new – or much improved – sensors.

It was not possible to find any studies to confirm or disprove the intuition that the number of sensors per mobile device is increasing. With that in mind, a quick study was conducted in order to investigate which sensors were more prevalent on mobile devices and try to spot any trends in the recent years.

Considering the group of mobile devices still available for purchase, six different sensors can be found on more than 50 % of such devices: accelerometer (79.5 %), Bluetooth (96.7 %), Global Positioning System (GPS) (75.6 %), proximity (60.9 %), radio (75.5 %), and Wireless Local Area Network (WLAN) (82.4 %).

There is an upward trend in the number of sensors per mobile device, with each of the identified sensors rising in usage (bar cases where overall presence was at or bellow 0.1 %). Contrasting mobile devices discontinued *versus* available for purchase, the percentage of units with zero sensors has dropped more than 30 p.p. into just 0.3 %. A more detailed overview of this analysis can be read on Appendix A.

It can be argued that some of the latest cars already have all of the sensors here presented, but that tends to be true only for vehicles in the high-end spectrum. Considering that nowadays every other person owns a smartphone [18], such devices have the potential to fill

in that gap for the vast amount of vehicles without sensory capabilities. This same argument was echoed by Chen et al. [19] and Fazeen et al. [20], both presenting research targeting the smartphone as a sensing device for the development of ADAS motivated by the cost associated with vehicles equipped with sensors.

Early works about smartphone sensing in vehicles have been categorized by Engelbrecht et al. [21] into four different groups, according to the type of information captured: traffic information, vehicle information, environmental information, and driver behavior information.

In the same study, the authors identified several advantages for the use of smartphones on vehicle monitoring systems. For one, instead of associating the collected information to a single vehicle, it ties the data to an individual – this is helpful for analyses related to a human being, rather than a vehicle, e.g., studies on driver behavior.

Another benefit is the decoupling of information about the vehicle, such as its age, make, or model, from the sensing solution. Using a smartphone for such system also reduces the cost of acquiring new, specialized equipment, all while providing connectivity to infrastructures outside the vehicle.

Finally, a simple but interesting convenience identified by said authors is the ability to use the smartphone for detecting phone usage during driving.

To counter, Engelbrecht et al. [21] went on to pinpoint some downsides of using these devices. Perhaps the most troublesome one is the limitation on battery power, which can become a hindrance if there's a need to collect data from a large number of sensors or run computationally intensive algorithms.

Having the mobile device and the vehicle in distinct coordinate systems is described as a difficulty, but this problem was already solved with satisfactory results multiple times by several authors [22–29]. Differences inherent to the broad range of existing smartphones can be mitigated and abstracted away with the use of application programming interfaces (APIs) provided by the different mobile operating systems (OSes).

The last challenge described by the authors is the "inaccurac[y] in cost-effective sensors used in smartphones" [21], which is the subject matter to be scrutinized on further sections of this thesis.

Pothole Patrol (P^2) [30] and Nericell [22] are the systems described in two of the most prominent papers related to the collection of environmental information. The main aspects of these two important works are discussed below.

With P^2, Eriksson et al. [30] produced one of the first road condition monitoring systems, using high-end accelerometer sensors operating at 380 Hz and GPS devices attached to a taxi probe car to collect data. This solution's hardware was composed by expensive, specialized parts and not analogous to capabilities of mobile devices available at the time.

In their architecture, a local personal computer (PC) analyzed the acquired data points for pothole detection and then sent the information via WiFi to a central server, where further processing occurred. Clustering and five different filters were applied in order to increase the precision of pothole detection. Data labeling was performed by a passenger inside the car, pressing a key on the local PC each time a road anomaly was felt, classifying it as one of the predetermined anomaly types. The road anomalies were determined before starting the experiments.

To decide the best placement for the accelerometer sensors, the authors conducted an experiment with units in three different places inside the car: attached to the dashboard, attached to the windshield, and attached to an on-board PC not firmly secured to the vehicle.

The presented results ruled out the accelerometer attached to the PC, as its output was too noisy when comparing it to the other two samples. Eriksson et al. [30] deemed the two remaining options as “quite similar”, despite their graphs showing the windshield unit to have a better signal-to-noise ratio (SNR). Notwithstanding, the dashboard position was chosen as it allowed the authors to install the sensors in the taxi without disturbing its passengers.

It is not clear what strategy was put in place to synchronize data from the accelerometers, the GPS unit, and the labeler’s inputs – this is a recurring theme in most prior art, as it will be possible to see. As Yi et al. [31] noted, the authors also failed to address how the thresholds for anomaly detection were chosen.

Developed by Mohan et al. [22], Nericell started to pave the way for the use of smartphones to monitor road and traffic conditions. Despite using two mobile devices for sound collection and mobile communications, accelerometer collection was handled by a SparkfFun WiTilt sporting a 3-axis accelerometer sensor capable of a sampling frequency of up to 610 Hz.

Throughout their study, Mohan et al. [22] described experiences as if they were using the smartphones to collect acceleration data at 310 Hz. However, the chapter describing the implementation reveals that data was sampled by the special-purpose WiTilt units and then sent it to the mobile devices via Bluetooth for further computation.

It is very likely that this did not have any impact on the results presented, but shows some bias on the authors’ part to try and present a solution whose novelty factor included the use of smartphones, which were starting to gain popularity at the time. In all fairness, it is important to keep in mind the strict hardware limitations of mobile devices available at the time. In later research, several authors [20, 32, 33] have shown identical systems using only smartphones sensors and arriving at similar conclusions.

With that being said, their analysis on determining accelerometer orientation is quite useful. Aforementioned authors propose a method to compute the Euler angles between the smartphones – or, the WiTilt accelerometers – and the vehicle. From this, it is possible to derive a rotation matrix and use it to reorient accelerometer data to the vehicle coordinate system. A validation of such data processing was performed, with good results being

presented: cross-correlation of two reoriented devices was similar to the cross-correlation between two devices similarly oriented.

Once again, Yi et al. [31] mentioned the shortcomings in explanations about thresholds' selection and Seraj et al. [34] pointed out the lack of clarification about the labeling technique used in Nericell.

Questions also remain on the chosen approach to synchronize data coming from different devices in order to determine the cross-correlation between two sources. It is certainly possible that some kind of manual synchronization was put in place, either at the time of recording or at the time of analysis. The former would be acceptable if the inherent accuracy loss was disclosed, but the latter would be inadequate without due diligence as manually tampering with the data can yield bias on drawn conclusions.[1]

2.3 DATA SYNCHRONIZATION AND STATISTICAL METHODS

For the purpose of comparing data acquired from multiple sensors, it is crucial to make sure that readings coming from all of them are synchronized – or at the very least to be conscious of existing skews.

As already stated in this section (vide page 7), prior research has commonly overlooked this problem. If in papers by Mohan et al. [22] and Eriksson et al. [30] this is simply not addressed, other authors either acknowledged synchronization issues or tried to mitigate them in diverse ways [26, 32, 35–38].

Tai et al. [32] manually shifted labels dictated by a motorcycle's driver to match the anomaly by inspecting a graph of the recorded accelerometer data. This is prone to human error and not a very good solution when dealing with sizeable datasets. A method by Tundo et al. [26] combined interpolation and shifting of data.

A similar proposal was made by Li et al. [35], where they simply maximize the correlation between two data streams, but this time automating the process. Albeit probably not being good enough, this would scale better for large amounts of data as long as an algorithm with less than $\mathrm{O}(n^2)$ time complexity was chosen to compute the maximization.

A better approach was presented by Paefgen et al. [36], where a system – comprised of a smartphone and an on-board diagnostics (OBD-II) device – was kept in sync using timestamps obtained via GPS. Despite the remarkable accuracy associated with the GPS system [39], smartphones tend to be equipped with cheap chipsets, commonly incapable of better sample rates than 1 Hz, fact that seriously dampers their reliability for keeping track of events in the order of tens to hundreds of milliseconds.

1 Several attempts to contact the authors were established through multiple communication channels. It was not possible to get timely clarifications on any of the presented issues.

Furthermore, it is important to note that current mobile devices do not feature real-time operating systems (RTOSes), so expectations on systems' response time should be adjusted accordingly. On this subject, Du et al. [37] developed a system to measure the International Roughness Index of pavement using accelerometers connected to a network of ZigBee modules. Albeit not being explicitly mentioned in this paper, these modules are capable of real-time communication, as prior art has shown [40].

Data from different accelerometers was compared using the timestamp collected from GPS units, with different frequencies used to sample data – 10 Hz for the accelerometers and 1 Hz for the GPS units. These rate differences can add up to significant drifts as the time of collection passes [41]. Despite the real-time approach of the above-mentioned architecture, this example serves to illustrate the fact that synchronization will hardly ever be perfect. Nevertheless, a number of measures can be put in place in order to achieve "good enough" levels of confidence – finding what is the appropriate level of confidence for each particular experiment is an exercise left for the reader.

A distinct approach found in prior research is to make use of statistical methods to compute the data read from different sensors to prepare it for feature extraction.

A simple option is to use linear interpolation to fill in the gaps of a dataset, be it data coming from an accelerometer sensor, positioning data from a GPS device, or speed provided via Controller Area Network bus [20, 26, 30, 42–45]. One of this method's main advantages is the ability to massage datasets of different lengths into a common one, making it possible to compute the correlation between them. Another convenient point is the possibility to apply linear interpolation to a very large number of data points without degrading performance. The lack of preciseness is a problem, with the error depending proportionally on the square of the distance between data points.

Polynomial interpolation is an alternative for producing a smoother result [28, 46], but it is not without its problems – it is computationally expensive if compared to its linear counterpart. This procedure is prone to presenting oscillatory artifacts, with especially high incidence at the edges of a data set – known as Runge's phenomenon.[2] A way to counter this is spline interpolation, which does not present such anomalies while still providing a smaller error and smoother result than linear interpolation.

Another way to tackle this is to use information about the trends embedded in the data series. The moving average[3] is used in prior works to compute a series of averages of different subsets of the full datasets [19, 23, 27, 31, 34, 47–49]. The size of these subsets

2 Named after its discoverer, Carl Runge, this phenomenon describes the observable oscillation on the edges of an interval when using polynomials of high degree for polynomial interpolation. This finding has shown that using higher degrees is not a sufficient condition to improve the accuracy of an interpolation.

3 Moving average is the prevalent terminology in prior research, but the same concept was also referred as moving mean, rolling average, rolling mean, rolling window, or running average. In at least one essay, multiple of these terms were used interchangeably along the text [34].

usually varies – even by a few orders of magnitude – from paper to paper. Cumulative and weighted variations of this technique exist.

For some use cases, like those pertaining to financial fields, this average is computed over a window of the previous *n* data points; but the majority of the references previously shown used an equal number of data from each side of the datum being computed – a technique commonly known as central moving average. Effectively, applying a moving average over a data series is a form of low-pass filtering; the output will be smoother, with noise being evened out and outliers being toned down. In the context of smartphone sensors, this characteristic is of extreme usefulness, as Almazan et al. [25] identified the high exposure to noise of such motion sensors as their major limitation.

A contrasting method of dealing with a number of the previously described problems is to use Dynamic Time Warping (DTW). Müller [50] defined it as a "technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions". Perhaps a clearer, shorter way to put it is that DTW aligns two time series by expanding and contracting the time dimension [51]. It has been useful in diverse domains – like bioinformatics, medicine, and entertainment [52–54] – but it is best known for publications in the area of speech recognition [55].

The best benefit DTW offers, as stated by Ratanamahatana and Keogh [56, 57], is the possibility to align, in a non-linear manner, two similar time series even if they are out of phase. In Figure 1, the vertical lines connect each point in one of the time series to another point in the other time series, resulting in a list of pairs of points minimizing the distance between those series. This list of pairs of points can be seen as a warping path and further used to transform each of the series in a way that minimizes the differences between them. Thus, one can assert that DTW provides an intuitive distance measurement, while always producing an optimal solution.

The fact that it produces an optimal solution leads to one of its downsides – its complexity. A naive approach to implementing DTW – such as using brute force – yields a solution with $\mathrm{O}(n!)$ complexity, but a clever use of dynamic programming can produce a $\mathrm{O}(n^2)$ time and space solution, as Müller [50] demonstrated.[4] As a result of the quadratic space complexity, using DTW becomes impractical when dealing with large volumes of data, with memory requirements in the order of a tebibyte (TiB) when handling time series with around $100\,000$ measurements [51]. Examples of previous studies using this technique in the context of smartphone sensors exist, especially in the area of driver behavior analysis [48, 58].

4 To be more precise, Müller [50] proved their solution to be $\mathrm{O}(nm)$, but this is expected to behave in a similar way to $\mathrm{O}(n^2)$ for n and m within identical orders of magnitude – note that this condition will generally be true when comparing datasets derived from smartphone sensors over a large period of time.

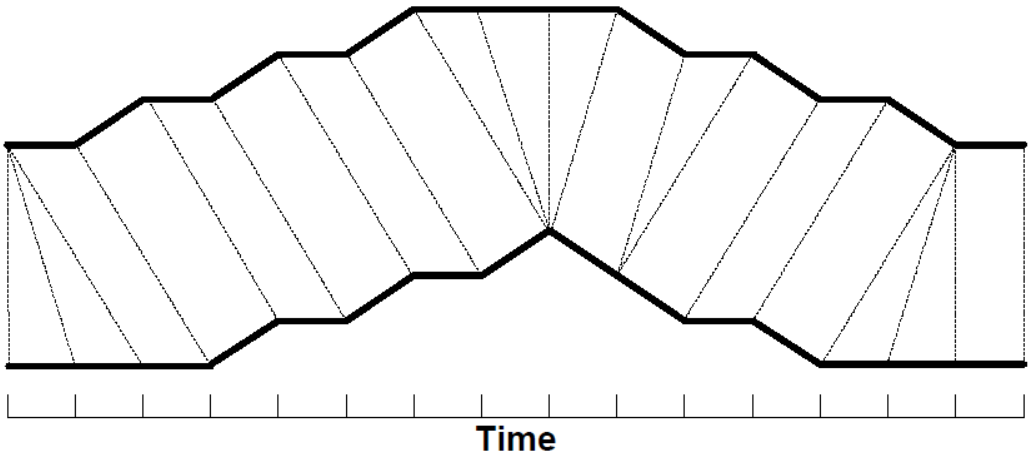


Figure 1: Representation of a warping path between two time series [ad. 51]. Both time series have similar values on the y-axis but one of them was shifted up to allow for a better visualization

Using DTW to warp two given time series makes it possible to derive the correlation between them [46], as they will always have an equal number of data points. Nonetheless, the alignment provided by DTW's algorithm is expected to increase the resulting correlation, introducing bias. Because of this, a parametric significance test is not valid and a randomization test should be performed instead [59].

Salvador and Chan [51] set to solve the problems DTW presents when working with big data, with a solution named FastDTW.[5,6] They envisioned an efficient DTW algorithm, with linear time and space complexity, while ensuring a nearly optimal warping path solution.

Despite conceding that Ratanamahatana and Keogh [56] had already demonstrated solutions with speed improvements allowing for practical analysis of a large number of time series at the same time, the objectives described by Salvador and Chan [51] were primarily focused on working with very long time series, that is, with a very high amount of data points.

Making use of an algorithm presented by Karypis et al. [60], authors developed a solution by way of three key operations: coarsening, projection, and refinement. This led to an implementation fulfilling all of their initial requirements of complexity, scaling well to long time series, and with an average error of $8.6\,\%$.[7] This outcome is orders of magnitude better than prior endeavors described by Ratanamahatana and Keogh [56].

5 Salvador and Chan [51] publicly released their Java implementation of FastDTW, with source code and documentation available at https://code.google.com/archive/p/fastdtw/

6 Dave Moten presented an enhanced version of FastDTW, refactoring the code from Salvador and Chan [51] to improve its extensibility and to ensure immutability of some critical objects – later this will be revealed as an important detail. Moten's source code lives at https://github.com/davidmoten/fastdtw

7 Results presented by Salvador and Chan [51] showed an average error of $8.6\,\%$ with the radius set to 1 when searching for the optimal warping path. Increasing this radius to 20 lowered the average error to $0.8\,\%$ and retained an acceptable execution time for the same amount of data

A key limitation identified by FastDTW's authors is the fact that it does not always find the optimal solution. This seems to be an acceptable concession to make considering the trade-off it provides – a solution fast enough to keep up with data being output at a very high rate from smartphone sensors.

3

EXPERIMENT PLANNING

Prior to executing the experiments, many decisions are taken during a planning phase. Those decisions and justifications behind them are described in this section.

After describing the context in which the experiments will occur, the hypotheses are formulated. From them, relevant dependent and independent variables are selected. Thereafter, subjects to represent the identified variables are discussed and chosen.

Then, the experiment design is characterized and the collection process is defined. Finally, analysis techniques are proposed and instrumentation developed for the purposes of experimentation is detailed.

3.1 CONTEXT PARAMETERS DEFINITION

Experiment planning should meet expectations set by this dissertation's objectives. A reasonable effort to mimic real world usage should also be carried out, to ensure that knowledge drawn can be used for practical products. Steps should be taken to identify the major constraints while taking the necessary steps to prevent them from being a risk to the experiments' validity.

Perhaps the most noticeable constraint is a geographical one. Resources available for the experiments will be provided by Bosch Car Multimedia, located in Braga. For this reason, they will occur on the roads of Braga, preferably in the areas around Bosch's plant or around the University of Minho, Campus of Gualtar. Empirical evidence suggests that such roads are good representatives of pavement quality found in other similarly-sized cities of Portugal.

Sensing the pavement quality of roads would be an example of an ADAS developed using smartphones sensors, so the experiment is developed in that context. This will also increase the SNR of collected data, since road anomalies can be labeled and then further analyzed.

As for constraints related to smartphones under analysis, they will be tested inside vehicles, held to the windshield by a car mount. This setup was a deliberate choice, to avoid introducing variables difficult to control.

For instance, experiments could be performed with the smartphone located in other parts of the car or – perhaps more interestingly – inside the pocket of the driver. Albeit not trivial,

the implementation of such a system is certainly possible [22, 24, 27, 29, 61]; however, doing so would make it far more difficult to perform a controlled collection of acceleration data, due to the possible variations of smartphone positioning inside the vehicle.

In fact, as a proof of concept, a small Android application was developed to test the possibility of such reorientation. Tests have shown that continuously collecting the vertical acceleration while accounting for the smartphone's ever changing orientation is not difficult. This prototype was eventually discarded for the already described concerns related to the quality of collected data.

3.2 HYPOTHESIS FORMULATION

In a real world scenario, any given smartphone is surrounded by a number of different variables possibly affecting its ability to accurately measure the acceleration of a vehicle. By means of observation, it is possible to propose a number of hypotheses to isolate those variables and study their impact on a smartphone's sensing capability.

Following are the hypotheses to be tested during the controlled experiments described in this document. For each different variable, two hypotheses are established – a null hypothesis and an alternative hypothesis. This is the method used in testing a statistical hypothesis and, particularly, when investigating a possible correlation.

Smartphones – and the inertial sensors embedded within – are very diverse, be it in size, materials, or software version. It is possible to anticipate that such differences might have a significant impact on the acceleration values those devices report, making them an obvious first candidate to be tested.

Hypothesis 1_0 *Using different smartphone models to record accelerometer data does not yield similar measurements of vertical acceleration*

Hypothesis 1_1 *Using different smartphone models to record accelerometer data yields similar measurements of vertical acceleration*

The car mount holding the smartphone influences the acceleration sensed by the smartphone since it acts as a proxy between the device and the vehicle where it is installed. This can be confirmed by contrasting the oscillatory movements of different car mount models while driving a vehicle. Therefore, it is important to investigate the effect they might have on the collection of such data.

Hypothesis 2_0 *Using different car mounts to hold the smartphone does not yield similar measurements of vertical acceleration by a smartphone*

Hypothesis 2_1 *Using different car mounts to hold the smartphone yields similar measurements of vertical acceleration by a smartphone*

It is not as easy to reason about the influence the sampling rate might have on the quality of information collected, but other authors have demonstrated the importance of this aspect [27, 34, 62]. Thus, there is a motivation for studying the consequence of varying a smartphone sensors' rate of sampling.

Hypothesis 3_0 *Setting different sample rates to acquire the data does not yield similar measurements of vertical acceleration by a smartphone*

Hypothesis 3_1 *Setting different sample rates to acquire the data yields similar measurements of vertical acceleration by a smartphone*

With even bigger complexity than smartphones, vehicles might have a great influence on the acceleration recorded by the smartphone. Differences in the levels of comfort experienced during a trip in different vehicle models are a good indicator of this effect. This prompts an assessment on the repercussions on the measurements caused by vehicles' diversity.

Hypothesis 4_0 *Using different vehicles to travel along an itinerary does not yield similar measurements of vertical acceleration by a smartphone*

Hypothesis 4_1 *Using different vehicles to travel along an itinerary yields similar measurements of vertical acceleration by a smartphone*

3.3 VARIABLES SELECTION

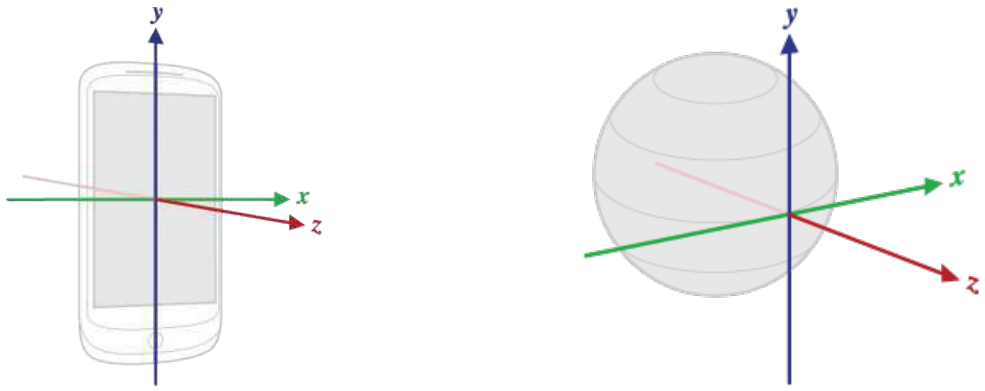
Both the dependent and independent variables emerge from a careful examination of the formulated hypotheses. Those variables will guide to what data should imperatively be collected to meet the research goals.

Extraneous variables were also identified and their possible impact on the experiment assessed. Additional data is considered for collection, despite not being directly related to the main objective. Nevertheless, one should collect just enough amount of data, as gathering arbitrarily large amounts of it may lead to an impractical processing time.

3.3.1 *Dependent variables*

Vertical acceleration Accelerometer data from each device is to be collected in the SI unit, metre per second squared ($\mathrm{m\,s^{-2}}$). Ideally, every device should be able to report data in such manner, as is the case of any Android device equipped with an accelerometer.

However, some devices might use other data representations, e.g., presenting it relative to the gravitational force (g). Such cases should be duly noted, with any calculation or approximation performed on the data being disclosed.



(a) Coordinate system (relative to the device) used by Android platform's Sensor API

(b) Coordinate system used by Android platform's rotation vector sensor

Figure 2: Android platform's coordinate systems [ad. Android API Guides]. By using a car mount to hold the device vertically, it is possible to align the device's y-axis in order to collect the vertical acceleration, represented in the geographical globe as the z-axis

An inspection of Figure 2 allows a better understanding of what vertical acceleration is, represented in Figure 2b by the z-axis. With the geographical globe as referential, vertical acceleration's axis points towards the sky and is perpendicular to the ground plane. To collect acceleration data, the sensing devices should be placed in a way that aligns one of its axes with the vertical acceleration – despite having other options available, like discussed in Section 3.1, this simplifies the data analysis.

For the particular case of Android devices, vertical acceleration data should be retrieved by means of an API abstracting the hardware[1] – either an accelerometer or an inertial measurement unit (IMU) – and reporting the sensed values back to the application.

3.3.2 *Independent variables*

Smartphone (inertial sensor) Due to the diversity of smartphones, it follows that the disparity of quality of the inertial sensors integrated into them can have an impact on the dependent variable.

Experiments should be performed with two different smartphones sporting distinct inertial sensors, preferably from different manufacturers. Ideally, those smartphones should be representative of two different price point categories to amplify differences in the quality of their components.

Car mount The level of vibration of different models of car mounts will be transferred, at least in some part, to the smartphone. It is reasonable to expect such difference in vibration to influence reported accelerometer data.

1 API detailed at https://developer.android.com/guide/topics/sensors/sensors_motion.html

A good litmus test to decide which car mounts to choose is to use two of them to hold a smartphone onto a windshield of a vehicle and manually try to displace them. If obvious differences are felt between two given car mount models, then those should be used in the experiment.

Rate of sampling Due to being more difficult to reason about the consequences of manipulating this variable without conducting initial experiments, the values chosen to perform the experiments should be reasonably supported by prior research.

Car Differences in the levels of comfort provided from different car models are a good indicator that they can have an impact on measured accelerometer data.

Experiments will be performed with at least two cars, preferably being good representatives of the Vehicles in Operation (VIO),[2] and having a significant difference in their price points and age.

Difficulties should be expected with this variable's subjects selection since the high cost associated with cars should have a significant impact on the number of available options.

3.3.3 *Other variables*

Each experiment testing a hypothesis will adjust just one of the described independent variables. Thus, for the experiments where they are not being modified, the remaining variables should be considered as controlled variables.

An initial study will be performed with the intent to establish the ground truth with a standard setup. Further results will be compared with this control group.

Prior research has demonstrated vertical acceleration recorded by an accelerometer being influenced by the vehicle's speed [20, 31, 37, 63]. Because of this, speed can be categorized as an extraneous variable.

Ideally, the speed of vehicles used in the experiments should be constant during the entire trip – making it a controlled variable. The driver of each vehicle will be responsible for maintaining the speed, but this is often difficult to accomplish, especially in urban areas.

With this in mind, speed will be monitored for the purpose of minimizing its impact on the experiments. Such data will be gathered in the SI unit of metre per second ($\mathrm{m\,s^{-1}}$) but, for convenience reasons, referred to in kilometre per hour ($\mathrm{km\,h^{-1}}$).

Data read from the gyroscope is not considered as an extraneous variable since it does not influence the dependent variable. Nevertheless, data from this sensor will also be collected

2 VIO represents the group of passenger vehicles currently registered in a country. A passenger vehicle is defined as a car or truck, used for passengers, excluding buses and trains

during the experiments for reasons beyond the scope of this dissertation. This rotational speed will be sampled in the SI unit, radian per second ($\mathrm{rad\,s^{-1}}$).

The collection of such additional data is not expected to have any impact on the studies to be performed but, for the sake of transparency, should be disclosed.

3.4 SUBJECTS SELECTION

The rationale leading to the choice of subjects to represent the variables described in Section 3.3 is documented in this section. Two subjects were selected for each independent variable, with one of them being used in the standard setup – the default one, so to speak. As for the vehicles' speed – the identified extraneous variable – possible values are discussed and a decision on one of them is taken.

Smartphone (inertial sensor) Three Nexus 5X are available to be used in this experiment. This model is fabricated by LG and incorporates a BMI160,[3] an IMU manufactured by Bosch, which outputs accelerometer and gyroscope data. This will be the smartphone use in the standard setup.

A Samsung Galaxy S Duos is also available for the experiments. Its accelerometer data is provided by an MPU-6000[4] from Invensense. This IMU outputs gyroscope data, too.

The 5X was released on 2015 and the Galaxy S Duos on 2012. Each runs a different version of Android as their OS. In what concerns to the retail price, GSM Arena[5] reports the former belonging to the mid-tier range and the latter as an entry-tier smartphone.

Due to their notorious differences in quality, price range, and year of release, these devices meet the requirements previously proposed.

Car mount Two iOttie Easy One Touch 3 will be used to hold the smartphones during the experiments. This was chosen as the car mount for the standard setup because empirical evidence has shown it to be very stable, not moving around too much even when driving on sections of pavement with multiple anomalies.

An unbranded car mount will be used to contrast with the above described. Again, empirical evidence demonstrated the questionable quality of this component, being especially unstable, wobbling a lot even when traveling on itineraries with good pavement conditions.

For more details, refer to Figure 11 (page 37) where both car mounts are depicted.

3 Details at https://www.bosch-sensortec.com/bst/products/all_products/bmi160
4 Details at https://store.invensense.com/ProductDetail/MPU6000-InvenSense/420595/
5 GSM Arena (http://www.gsmarena.com/) is a web site specialized in gathering and listing mobile devices specifications.

Table 1: Distance between consecutive accelerometer measurements at different speeds for different systems, contrasting with the sampling rates chosen for this dissertation [ad. 34, minor inaccuracies corrected]

System	rate (Hz)	distance (cm) traveling at		
		$25\,km\,h^{-1}$	$50\,km\,h^{-1}$	$75\,km\,h^{-1}$
P² [30]	380	1.8	3.7	5.5
Nericell [22]	310	2.2	4.5	6.7
RoADS [34]	93	7.5	14.9	22.4
Pertunnen [33]	38	18.3	36.6	54.8
Tai [32]	25	27.8	55.6	83.3
Standard setup	200	3.5	6.9	10.4
Alternative setup	50	13.9	27.8	41.7

Rate of sampling The choice of sampling rate for the standard setup was quite pragmatic. Both chosen smartphones reported being capable of sampling data at 200 Hz (reading data each 5 ms), so that was the value selected. This is the minimum delay at which each chosen inertial sensor is able to operate.

In a study regarding road roughness condition, Douangphachanh and Oneyama [63] proposed the frequency range of 40 Hz to 50 Hz as the best solution to sample smartphone acceleration sensors. Supported in their study, the rate of 50 Hz (each 20 ms) will be used to perform a comparison.

Table 1 summarizes how the chosen rates of sampling compare to related studies found in prior research. The rate chosen as standard (200 Hz) falls short only to systems from P² and Nericell, where special purpose accelerometers were used instead of smartphone sensors. As for the alternative rate (50 Hz), it is in line with other smartphone-based systems and has the benefit of producing a smaller amount of data.

Car Two cars will be available to perform the experiments. The first is a Mazda 3 from 2007,[6] which was chosen to be part of the standard setup because it was the only one available at all times. The second car is a Volkswagen Polo from 2016, a rented car available to the researchers only during a single day.

Despite belonging to the same segment and similar price points, these two vehicles have a nine-year gap between them. Because of this, some differences in the sensed vertical acceleration are expected, both due to the aging of multiple parts and improvements in their quality.

6 The Mazda 3 is known as Mazda Axela in Japan and China

Speed As stated in Section 3.3.3, the speed at which the vehicles travel is considered as an extraneous variable to the experiments. To minimize its impact on the dependent variable, experiments will try to maintain the vehicles' speed at $30\,\mathrm{km\,h^{-1}}$.

This is a reasonable value for two reasons. Driving at even lower speed would certainly make it easier to keep it steady but would be prone to traffic congestion. Moreover, driving faster would only be possible up to $50\,\mathrm{km\,h^{-1}}$, above that would be considered as speeding in most of the itinerary where experiments are to be performed.

As discussed throughout the introductory chapters, traveling at such speed would mean that collected acceleration data could later be analyzed to identify road anomalies as small as $4.2\,\mathrm{cm}$ (with the standard setup). Table 1 contrasts this value with systems developed in previous research, with several possible speeds for reference.

3.5 EXPERIMENT DESIGN

During an experiment, a vehicle is used to perform a set of maneuvers on a predefined itinerary to capture data within a city environment. This vehicle is equipped with a number of Android smartphones, each running an application created for this purpose. A car mount is used to keep the smartphone stable while the vehicle travels.

The Android application has capabilities to acquire, present, and export sensors data from the smartphone where it is running. This application collects data from the accelerometer, gyroscope, GPS coordinates, and speed. To annotate the experiment, a co-driver uses a second Android application, capable of storing the type of anomaly detected and a timestamp of its occurrence.

Each experiment will test one hypothesis with two setup configurations being used. One of those configurations remains the same across every experiment, working as a control setup: the same smartphone, car mount, rate of sampling, and car. The alternative setup changes only one of those variables. With the goal of increasing the results' statistical significance, every experiment will be performed for a total of five times.

To ensure a rich diversity of pavement anomalies to be detected on the experiments, a survey of potential itineraries was performed in the roads of Braga. In the identification of these potential itineraries, it was taken into account the total number of pavement anomalies, the number of different types of anomalies,[7] the itinerary's size, and the possibility to make a full travel maintaining the vehicle's speed.

The itinerary represented on Figure 3 was chosen from a group of identified candidates. This path satisfies all of the specified requirements because each type of anomaly occurs at least once, there are sections of asphalt and cobblestone, it takes approximately five minutes to complete it, and it is fairly easy to stabilize a vehicle's speed in most of its sections.

7 Section 3.6 has more information on the types of pavement anomalies identified

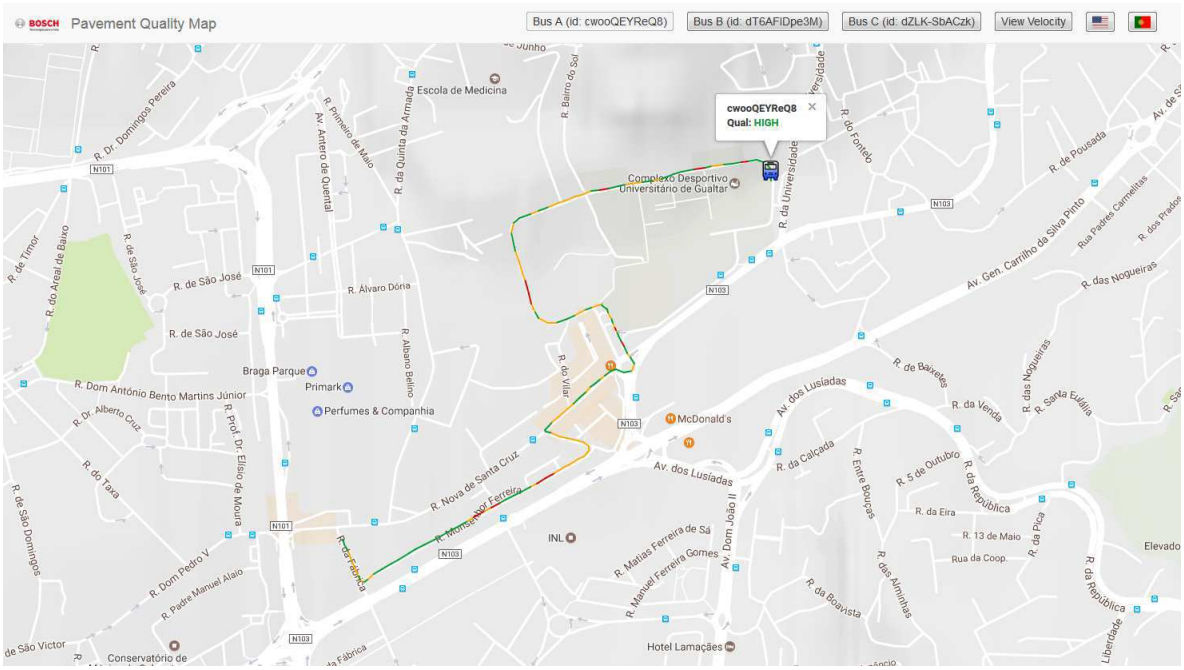


Figure 3: [Best viewed in color.] Chosen itinerary for performing the experiments, near the University of Minho, Campus of Gualtar. Starting point on the left and finishing position on the right. Screen capture taken from a prototype application plotting pavement quality information on a map based on accelerometer data collected during the experiments

When designing their experimental plan, Mednis et al. [23] expressed the need to perform all of their experiments on a short time span – ideally in a single day – because "[s]uch approach ensured minimal road changes between the data acquisition activities" [23].

In the research developed by Mednis et al. [23], it was important for all the data to represent a similar snapshot of the roads being studied. In opposition, one can assert that this is not the case for the work described in this dissertation, as data will only be compared within each field study.

It is surely advisable to complete all repetitions of a given experiment in a single day, but studying different independent variables in different days is not expected to have an impact on the results' validity.

3.6 COLLECTION PROCESS DEFINITION

Experimental data will be collected by a team of researchers from late months of 2016 to early months of 2017, in an itinerary in the streets of Braga, near the University of Minho, Campus of Gualtar. Refer to Figure 3 for a more detailed view of the traveling course planned for the experiments.

The said team of researchers is composed by members of Bosch's Cloud Applications team, where the author is integrated. The number of researchers assigned to each experiment will vary with the needs identified.

Due to researchers' scheduling constraints, experiments will be performed during business hours. With this restriction in mind, and to minimize the impact of traffic on the results, experiments will occur during periods less prone to traffic congestion, like the middle of the morning or middle of the afternoon.

Every repetition of an experiment will start with the vehicle not moving but having its engine running for recording 5 seconds of accelerometer data. This will allow the collection of reference accelerometer values representing the noise caused by the vehicle's engine. Such reference values can then be used to calibrate the smartphone's accelerometer, but are otherwise considered used for the purposes of the experiment.

Upon the completion of the initial phase, the Android application starts collecting and storing smartphone sensors data. The researcher in the co-driver position uses the annotations application to mark the start of a recording session and command the driver to start driving the vehicle according to the driving plan.

While the vehicle is moving, the co-driver is responsible for making annotations of the predetermined pavement anomalies as they are experienced along the route, using an Android application developed for the purpose. The driver, always trying to keep a constant speed, is responsible for driving through the road without avoiding the anomalies.

Road anomalies identified in the chosen itinerary are classified as either eroded asphalt, long bump, short bump, manhole, or pothole. Transitions between asphalt and cobblestone are also annotated.

These annotations will later aid in performing data validation, e.g., by using a graph to visually confirm significant differences in the sensed acceleration while driving through a pothole.

Reaching the finishing position, the driver stops the vehicle and, after that, the co-driver uses the annotations application to label the end of the recording session. Then, the smartphone with the sensors data application is instructed to stop the collection of data.

Meanwhile, during the whole experiment execution and in parallel with the collection phase, the application will be sending data to an endpoint by means of a 4G connection. Within 5 seconds of stopping data collection, the application presents a notification to inform the team of researchers that all data was successfully synchronized with the endpoint. This message of success marks the recording session as a valid one.

Finally, if there are more repetitions of the experiment to perform, the team of researchers moves to the starting point of the itinerary and restart the procedure described in this section.

3.7 ANALYSIS TECHNIQUES

A suitable method to test the hypotheses formulated on Section 3.2 is to compute the sample correlation coefficient between vertical acceleration collected by the pair of smartphones used in each experiment. This correlation coefficient will determine the similarity of reported accelerometer data from distinct devices and how strong that similarity is.

The sample correlation coefficient between two time series, x_i and y_i, is defined as

$$r = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}$$

with $\bar{x}$ and $\bar{y}$ referring to the mean of each time series. The equation yields a normalized result varying between -1 (inversely correlated)[8] and 1 (perfectly correlated). In the case of the two time series being entirely uncorrelated, $r = 0$.

Multiple difficulties are anticipated in using this technique. For one, correlation between raw data is expected to be very low due the noise associated with measurements provided by IMUs embedded in smartphones.[9]

Also, because Android is not an RTOS, it is difficult to ensure that two different measurements happened exactly at the same time. For this reason, it is very likely that the two datasets fall out-of-sync.

Lastly, as a close examination of the presented equation can reveal, an equal number of data points for both time series is an imperative to compute the correlation between those datasets under analysis.

To address the described drawbacks, techniques for processing the data were studied and tested to be further used. One of such methods was the central moving average, which has the advantage to even out the noise on the collected vertical acceleration and slightly reduces the effect of not having the data points synchronized. On the other hand, the moving average is not a satisfactory solution in the event of having two data sets with a different length – like it is the case when using different sampling rates.

DTW aims to solve the problem of data sets having different lengths and being out-of-sync, while also reasonably dealing with noise. Thus, the two time series under analysis will become aligned, even if they were out of phase. A downside of this choice is the need to develop specific tooling to automate the correlation computation, as related existing tools are not tailored to the needs of this dissertation's experiments.

8 An inverse correlation means that when one of the time series goes up, the other goes down
9 Increasing the SNR of such measurements is a motivation to drive through road anomalies on purpose

Given the tendency of DTW to bias the correlation for higher values, it is not valid to use a parametric significance test [59]. Alternatively, a randomization significance test will be performed.[10] In this test, a large number of surrogates of each dataset will be prepared by randomizing the order of the data points. Then, each of these pairs of copies will be correlated with each other and their results plotted for visual inspection.

The original correlation coefficient will be statistically valid if and only if it is significantly different from the correlation values of the surrogates, i.e., *iff* it is at the tails of the sample distribution of correlation coefficients, formed by rank ordering the computed values for each pair.

If valid, the correlation coefficient will then be compared to a baseline, set by an initial experiment with a control setup to draw conclusions on the impact of the independent variable under scrutiny. Such comparisons will finally allow making decisions about the hypotheses previously presented.

3.8 INSTRUMENTATION

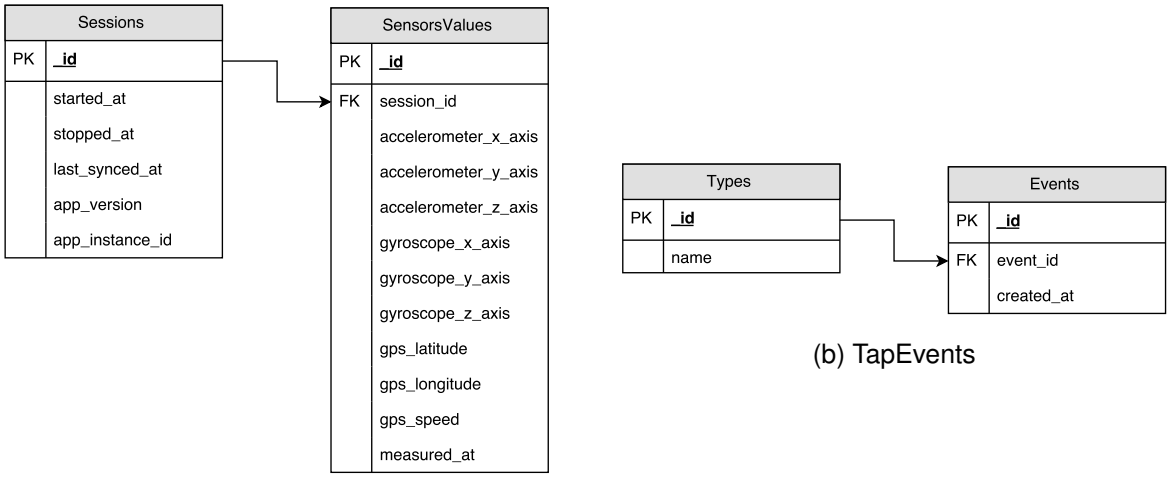
To assist the operation during the experiment execution and data analysis, three special-purpose tools were identified as in need for development.

Smartphones used in the experiments needed an Android application to collect and export their sensors data. After testing existing applications with similar features it was concluded that none of them satisfied all of the elicited requirements. So, to fulfill those requirements one such application was developed – an Android application named *Bumpr*.

A second smartphone application is needed to assist the researcher's job of annotating road anomalies, beginning, and end of recording sessions. With the number of features being rather low, the development of this application – *TapEvents* – was mostly focused on non-functional requirements, namely, on building an efficient user interface (UI) that could be used while navigating through the itinerary.

To automate the data analysis phase, a desktop application was developed to, first, compute the correlation coefficients of collected vertical acceleration data and, then, statistically validate the results. This application, *TimeWarper*, makes use of an open implementation of the DTW algorithm (FastDTW) to prepare the streams of sensors data for analysis. Most of the development effort for this application was spent on maximizing its parallelism since it had to produce results in an adequate time frame.

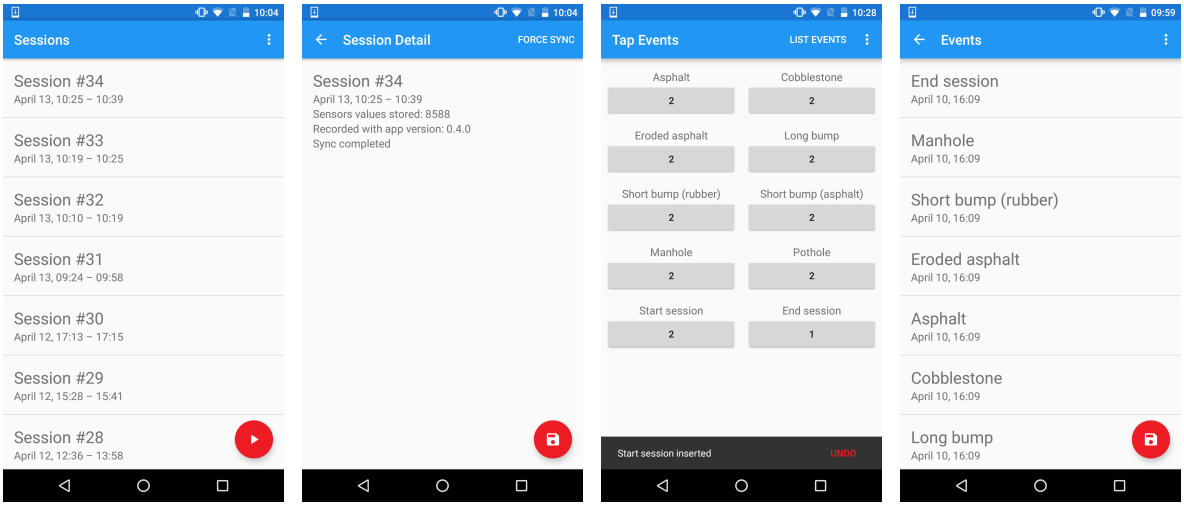
10 Random Shuffle (RS) [64] will be used for this significance test. Schreiber and Schmitz [65] list Random Phases (also known as Fourier Transform) and Iterative Amplitude Adjusted Fourier Transform as alternative algorithms



(a) Bumpr

(b) TapEvents

Figure 4: Android applications' relational models



(a) Bumpr main UI, listing all recording sessions and a button to start recording

(b) Bumpr secondary UI, showing details about one session and a button to export its data

(c) TapEvents main UI, displaying buttons to log session activities and a temporary snackbar to undo the last action

(d) TapEvents secondary UI, listing all logged session activities and a button to export all data

Figure 5: Screenshots of Android applications' UIs running on a Nexus 5X

3.8.1 *Bumpr*

Bumpr has a Model–View–Presenter (MVP) architecture,[11] with some interaction with the Android platform's Services API and Loaders API. On the Android platform, a service is an application component designed to perform long-running operations in the background without providing a UI; a loader can be used to implement an observer monitoring the data source for changes and refreshing the UI, freeing the main thread and improving the application's responsiveness. Functionalities like sampling the sensors, storing data in a database, or network communication are all handled by specialized services.

This application's relational model is presented in Figure 4a. Recording sessions have an initial and a final timestamp, as well as the last time its information was successfully sent to an endpoint; information related to the application's version and a unique device identifier was also stored. Associated with a session, the smartphone collects and stores data from the accelerometer (3 axes), gyroscope (3 axes), GPS coordinates, vehicle's speed, and a timestamp. All sensors are sampled at the same rate and values reported by them are stored as a tuple.[12,13]

Bellow is a brief overview of the most relevant requirements identified for this application, roughly ordered by their importance (refer to Appendix B for a comprehensive description of them all):

- Device starts and stops reading and recording sensors data
- Device automatically stores sensed data on itself
- Device calibrates its sensors before starting to record session data
- Device sends sensed data in near real-time to an endpoint
- Researcher uses the device to see a list of the recording sessions
- Researcher uses the device to see details about a recording session
- Researcher uses device to configure which sensors to record and the corresponding rate of sampling

11 Both MVP and Model–View–Controller (MVC) propose solutions to the same problem, with the key difference being that in MVP the view and model do not communicate directly, but through a presenter – Fowler [66] calls it a Supervising Controller. Compare this to a traditional MVC approach, where view and model might exchange information without relying on the controller

12 This means that sensors with lower frequency might have repeating values in consecutive rows. For the smartphones operated in the experiments described in Chapter 4, this would only happen for values reported by the GPS which was capable of outputting data from 1 Hz to 10 Hz. Location and speed data are only collected for sanity checking the accelerometer data, so a lower refresh rate is an acceptable trade-off

13 Consecutive rows with completely repeated sensors values (apart from the primary key) can occasionally occur. This happens due to halts in processing caused by garbage collection (GC). Experimentation has shown this to happen at most once per minute, lasting for no longer than 50 ms. GC's period is mainly affected by the allocation of a large number of objects, so a pooling strategy was put in place to avoid creating too much of them. However, to avoid losing precision, immutable classes like `BigDecimal` were chosen to hold sensors values; their immutable nature precluded the pooling strategy for such objects. Trading a very small percentage of repeated values for a higher precision during the whole collection process was a conscious choice, yielding better results for virtually no cost

Most of these functionalities are implemented as a service component and, thus, have no UI. Bumpr's main UI shows a reverse chronology of all recording sessions, information about when a session has started and stopped, and a button to start a new session or stop the ongoing one (see Figure 5a).

Tapping on any session leads to a detailed view of it, with the number of sensors values collected during that session, information related to communication with the endpoint, and a button to optionally export data in the comma-separated values (CSV) format (see Figure 5b).

Settings can be tuned in a tertiary view allowing, e.g., to choose the rate of sampling or to input the endpoint's domain.

Special care is required when handling accelerometer data from the smartphones. The Motion Sensors APIs provided by the Android platform allows the collection of raw acceleration data as sensed by an acceleration sensor. This sensor determines the acceleration applied to the device (A_d) by measuring the forces applied to itself (F_s) following the relation:

$$A_d = -\sum \frac{F_s}{\text{mass}}$$

Since the force of gravity (g) is always affecting the measured acceleration, this relation can also be described as:

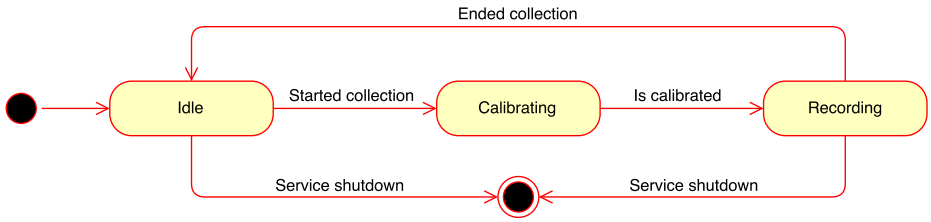
$$A_d = -g - \sum \frac{F}{\text{mass}}$$

From this, it follows that a device vertically held in a car mount on a stopped vehicle will report its vertical acceleration as $g = 9.81\,\mathrm{m\,s^{-2}}$.

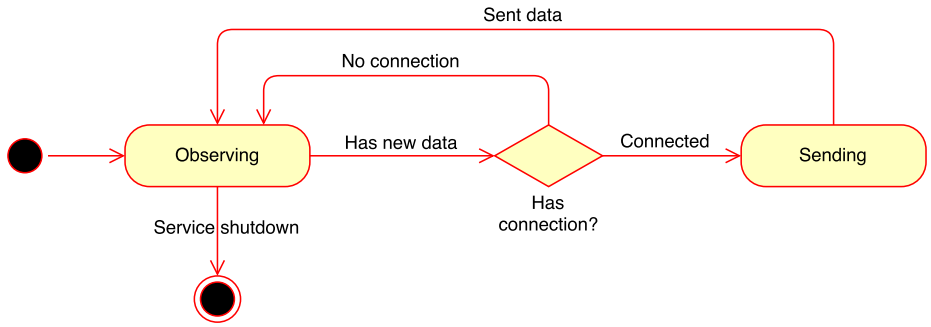
Thus, to obtain the intended data, a calibration must be performed to remove the force of gravity – this is usually known as linear acceleration. Several techniques can be used to filter the unwanted gravity, in Bumpr acceleration values are sampled during a brief window of time, then processed to remove outliers, and finally a reference to the average value for each axis is kept as an offset. After that, when storing sensed acceleration values, each axis is subtracted its offset – this technique is valid as long as the positioning of the device is maintained during the recording session.

Bumpr has two core behaviors: data collection and data export. These are always present during the application's lifetime and executed in parallel. They have loose coupling since their only connection is the production and consumption of sensors data.

Figure 6a shows the state diagram associated with data collection. On initialization, the device remains *idle*, until it is instructed to start the collection, switching to *calibrating*. After finishing the calibration phase, already described above, it starts *recording* the sensors data. This continues until an instruction to end the collection of data is made, returning to the *idle*

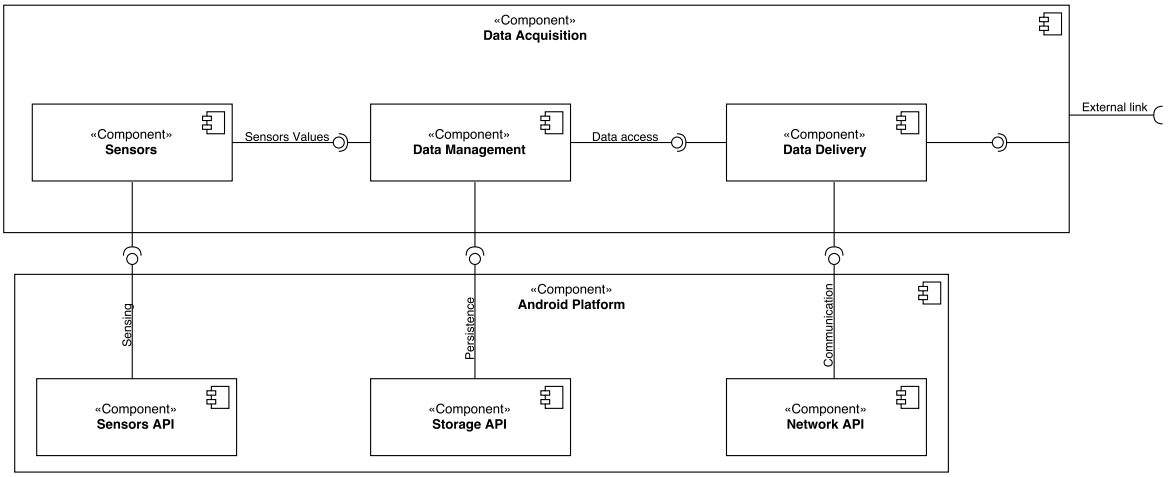


(a) Data collection state diagram



(b) Data export state diagram

Figure 6: State diagrams describing the behavior of Android application Bumpr during data collection and data export



(a) Data Acquisition component diagram and its dependencies on the Android Platform

(b) High-level component diagram, detailing the relationships between Data Acquisition and other components of the system

Figure 7: Component diagrams detailing the Data Acquisition and its relationships with other components

state. The final state should only occur when the device is *idle* and instructed to terminate, but abnormal termination might happen when it is *recording*.

The state diagram of data export is presented in Figure 6b. On initialization, the device starts *observing* the sensors data stored but not yet exported. When new data is detected, it asks the Android platform if there is connectivity available: if not, it continues *observing*; if there is, it connects to an endpoint and starts *sending* the new data. After all data has been sent, it returns to an *observing* state. The final state should only occur when the device is *observing* and instructed to terminate.

The Data Acquisition component (see Figure 7a) is an abstraction of the Bumpr application described in this section. It encompasses three smaller components: Sensors, Data Management, and Data Delivery.

Sensors component is responsible for sampling the sensors, calibrating the sensed values, and pushing data to the Data Management component. This component interacts with the Android Platform's Sensors API which abstracts the physical sensors.

Data Management component is a thin layer providing specialized interfaces to persist and access acquired data. In addition to intermediating Sensors and Data Delivery components, it is capable of communicating with the Storage API provided by the Android Platform. The data is locally stored in a SQLite database closely tied to the application but is considered as being outside the Data Acquisition component since most data persistence responsibilities are relegated to the Android Platform.

Finally, Data Delivery component accesses and processes persisted data before establishing a Hyper Text Transfer Protocol Secure (HTTPS) connection to an endpoint exposing representational state transfer (REST) APIs. Acquired data is securely delivered through that connection, where it will be persisted for later use, i.e., for data analysis. To communicate with the outside world, this component collaborates with the Android Platform's Network API.

This leads to the overview of the entire pipeline where sensors data flows, as Figure 7b shows. A thorough explanation of the remaining components – AppBackend and Service – is outside this dissertation's scope, but a brief summary follows.

The AppBackend component provides REST APIs, accessed via HTTPS, to persist and access sensors data by way of a single endpoint. For instance, data retrieval for the analysis described in Chapter 5 is performed by way of this component instead of going through every smartphone used in the experiments. It is also responsible for processing the sensors values for multiple uses in the Service component.

The Service component provides a visual representation of information extracted from data collected by the smartphones, laid out in a map interface (for a glimpse of this see Figure 3, page 21).

3.8.2 *TapEvents*

Despite its simplicity, TapEvents' architecture is also based on MVP, collaborating with the Android platform's Loaders API. It comprises two features to aid in data collection during the experiments execution.

One of such features allows a researcher to annotate road anomalies as they happen during the experiments. During the development of this functionality, its UI was tuned several times to improve the application's usability. For instance, buttons were enlarged to provide a bigger input area, the number of occurrences of each event was added to the button to visually confirm a successful insertion, and a snackbar was added allowing to undo the last action performed (see Figure 5c).

The other functionality is related to exporting the annotations to a file. A reverse chronology of all annotated events is presented alongside a button to export the data after choosing a file name. The UI is not very complex (see Figure 5d) and all of the hard work is performed in the background. When exporting the annotations, stored data is queried and then converted to the CSV format, ready to be used in the data analysis phase.

The relational model used within this application is shown in Figure 4b. Types of events (see Section 3.6 and Figure 5c) were elicited prior to the implementation and then hardcoded in the application.[14] Every time a user logged an event, a tuple composed of the type of event and a timestamp was created.

3.8.3 *TimeWarper*

TimeWarper is a Java application built to automate the data analysis phase. Source code for this tool can be found in Appendix C.

For each experiment, it iterates over every valid recording session to compute the correlation coefficient between vertical acceleration data recorded by the two smartphones used. To do this, it starts by computing the warping path between the two time series. It does so by using FastDTW [51],[15] an open implementation of the DTW algorithm.

The result is a warping path which can be used to effectively warp each time series. After that, it is possible to compute the correlation coefficients using the algorithm described in Section 3.7.

14 It would be reasonable to provide a create, read, update, and delete (CRUD) interface for the types of events, but such effort was deemed unnecessary since no modifications were anticipated. Moreover, if needed, the cost of change would be very low – the problem would probably be solved with a one-liner migration adding a new record to a table

15 Dave Moten provided a refactored version of FastDTW (see page 11 for source code reference), improving its extensibility and ensuring immutability of critical objects. Since TimeWarper aimed to maximize parallel computation, this was the chosen version to the detriment of the one provided by Salvador and Chan [51]

To statistically validate each correlation coefficient, a hundred copies of each pair of time series is created – the surrogates. Then, an RS algorithm is applied to each surrogate pair (again as described in Section 3.7). Now each of these pairs of surrogates will pass trough the same process described above: warped path is computed, surrogates are warped, and then correlation coefficient is computed.

Finally, a log file is produced identifying the experiment and recording session. Inside this file are all correlation coefficients computed, by increasing order, with the original pair's coefficient being highlighted for better identification. This log file would ultimately be used for the data analysis described in Chapter 5.

4

EXPERIMENT EXECUTION

With all set, we began doing the field studies, always following the procedure described in Section 3.5. These experiments happened in a time span of three months, in contingency with the availability of hardware resources and the readiness of software used to collect data.

A simple tryout was scheduled to validate the planned experiment design previously defined, in what could be described as a meta-analysis. This experiment dry-run was performed on October 13th, 2016 in a section of the previously selected itinerary with the objective to detect and correct unforeseen mistakes during the planning phase.

The first run was carried out on November 2nd, 2016 aiming to set a ground truth to subsequent field studies. In this analysis, a setup was built in a single car with two similar smartphones, as well as similar holders and sample recording rates. Information gathered from this experiment would act as a control group, setting the baseline against which future runs would compare.

Second and third runs were both executed on November 23rd, 2016. An appropriate hardware arrangement was chosen to allow two independent variables to be tested at the same time – two different smartphones and two different holders for, respectively, the second and the third experiment execution. Further details on this hardware layout can be read and observed in Sections 4.2 and 4.3.

Finally, on January 13th, 2017, fourth and fifth runs occurred, for testing different sample rates and different cars, respectively. Once again, equipment was selected in such way to support running two experiments in parallel, with extra details on this deferred to Sections 4.4 and 4.5.

The following sections present additional details about each performed field study.

Terminology This chapter makes use of two terms: *run* and *session*. A run refers to an instance of a field study where an experiment is being conducted. A session – short for *recording session* – is the time window delimited by the start and end of a driving exercise, during which sensors data is being recorded. Each run aggregates a number of sessions.

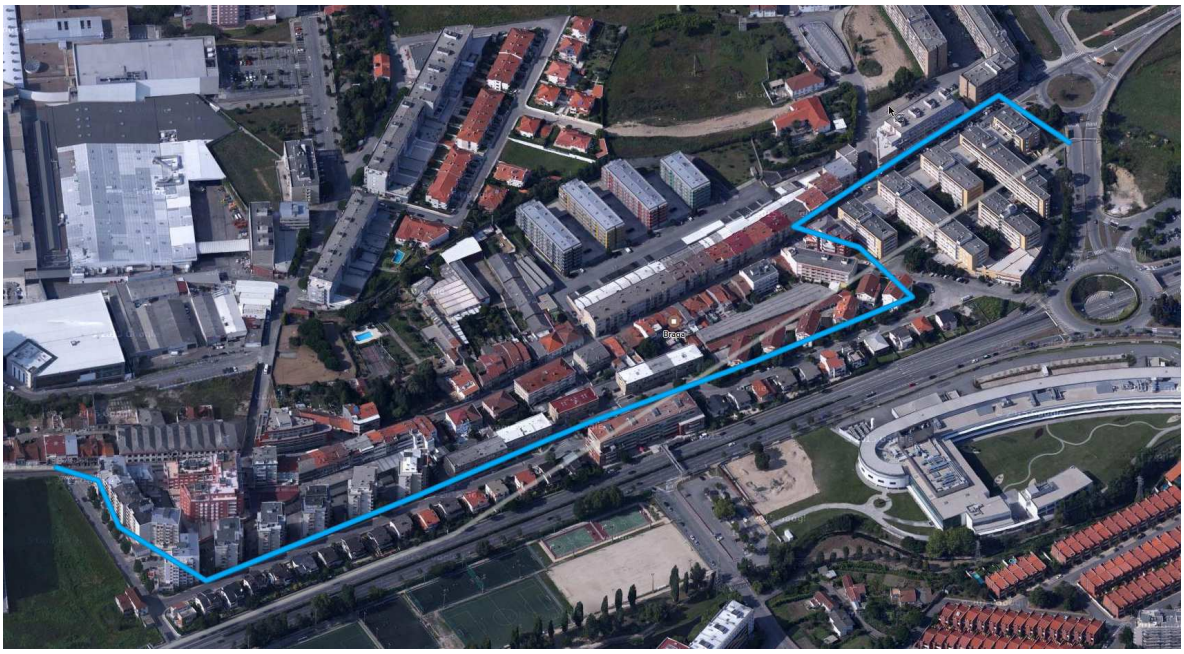


Figure 8: [Best viewed in color.] Segment of the chosen itinerary traveled in the experiment dry-run, depicted in blue. Starting point on the left and finishing position on the right

EXPERIMENT DRY-RUN

In order to validate our experiment design and collection process – while identifying and correcting previously unforeseen problems – we selected a section of the chosen itinerary to perform a rehearsal experiment. The itinerary's segment traveled in this dry-run is represented in Figure 8.

Two smartphones were used during this tryout: one for recording accelerometer data, held by a car mount; and another running the annotations application, handled by the co-driver. In addition to taking annotations about the road surface, the co-driver was also responsible for taking notes about all relevant details identified by both researchers during the trip. The driver was responsible for keeping the vehicle at a constant speed and for going through all chosen road anomalies. Both researchers kept constant communication, exchanging their impressions on the exercise.

As a result, a number misconceptions were identified in this dry-run and promptly fixed in the following days. For instance, the application acquiring sensors data was recording GPS coordinates with an insufficient level of accuracy, which became evident after projecting the data points in a mapping interface. Also, the application for annotating the pavement anomalies required some work on the user interface, especially with the addition of an option to undo the last action performed on it.

It also became clear that we should write down in what order we were approaching pavement anomalies as well as their approximate location, or it would be very difficult to make good annotations. Furthermore, the need to have a third researcher on the vehicle was identified whose responsibility would be to aid the navigation of both the driver and co-driver by reading the list of road anomalies ahead.

The amount of time spent in creating a data spreadsheet to analyze the collected data lead to the decision to create a more streamlined workflow, with most of the steps being performed programmatically. Spreadsheets would still be used to visually inspect the data, but some tools needed to be built to perform more complex actions like merging the sensors data with the annotations or computing the correlation between two data sets.

During this preliminary exercise, a major concern became obvious – timestamps from the smartphone used to collect data were not synchronized with those produced by the smartphone used to label road anomalies. Some discrepancies were expected – since Android is not a an RTOS – but empirical evidence has shown divergences in the order of tens to hundreds of seconds. Several solutions were investigated with the intent to correct – or, at least, assuage – this problem.

On the positive side, it was possible to validate the data collection pipeline's entire operation. The smartphone application stored the data without issues and was able to exfiltrate all of it to the dedicated server via a mobile connection. This dedicated server's availability was never interrupted.

In hindsight, it is easy to recognize the importance of this experiment dry-run, as it helped to anticipate and mitigate a large number of possible difficulties and made it possible to effortlessly perform all of the subsequent runs.

Due to the ad hoc nature of the procedure here described, sensors data recorded therein was not considered for further analysis. Nevertheless, a copy of such information was stored for future reference in case the need to check it ever arose.

4.1 FIRST RUN

The first run's goal was to set a ground truth and use it as a standard for comparing the results from other experiments.

A symmetric configuration[1] was prepared to accomplish this objective: two Nexus 5X, incorporating each a Bosch BMI160 accelerometer, running the same OS version, with the same recording application version sampling at 200 Hz, mounted on similar iOttie Easy One Touch 3 in identical positions and angles, and inside a single 2007 Mazda 3. Figure 9 is a depiction of this symmetric setup, a photo taken during the experiment.

1 By *symmetric configuration* or *symmetric setup*, one means to say that there was a pair of similar hardware and software for every item used in this particular experience

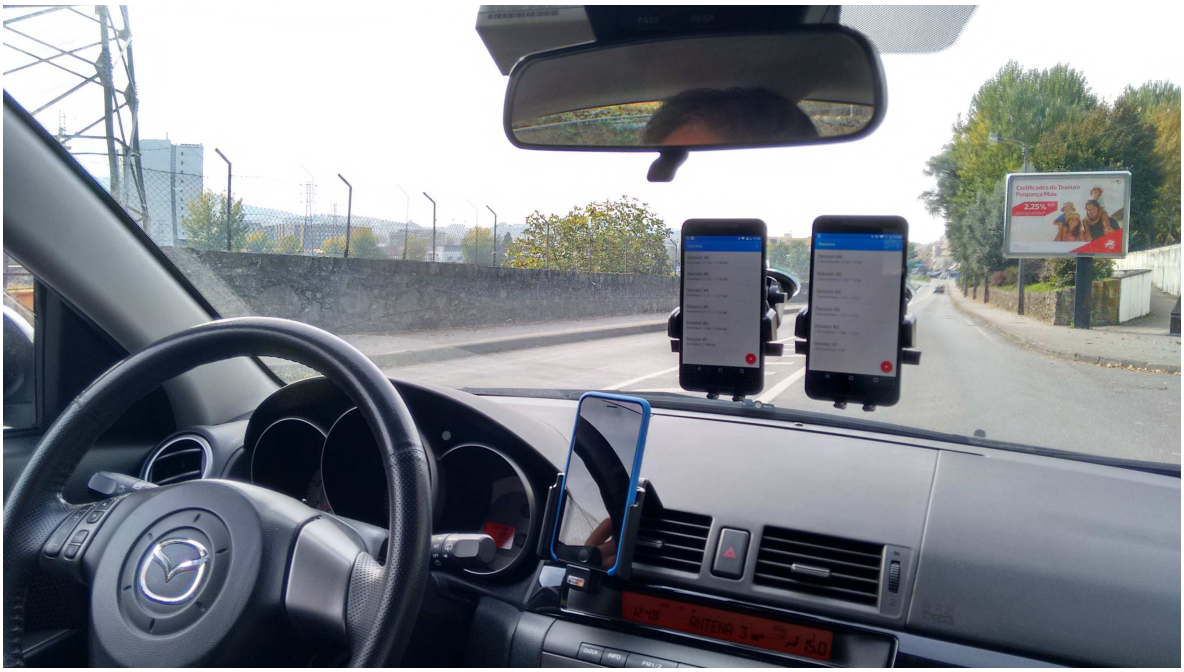


Figure 9: Equipment setup for the first run, a symmetric configuration with two similar smartphones running the same software versions, mounted on similar car mounts in similar positions and location, inside a vehicle

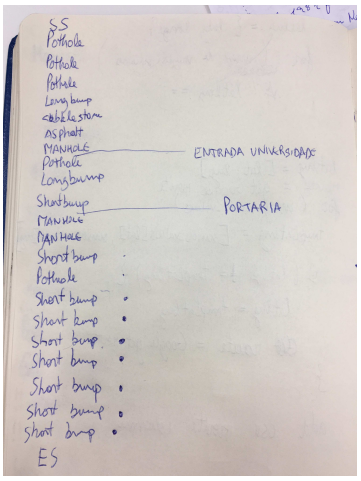
Leveraging lessons learned in the dry-run, this time there were no major issues identified. In addition to small adjustments, the experiment was performed by three researchers with different responsibilities, an events list was prepared in advance, the recording application was fixed to increase the precision of GPS coordinates, and the annotations application was tweaked to improve its user interface and have an *undo* option.

In spite of this, the chosen time of day to perform it was not the best, with some traffic congestion making it difficult to maintain the vehicle's speed during the entire itinerary.

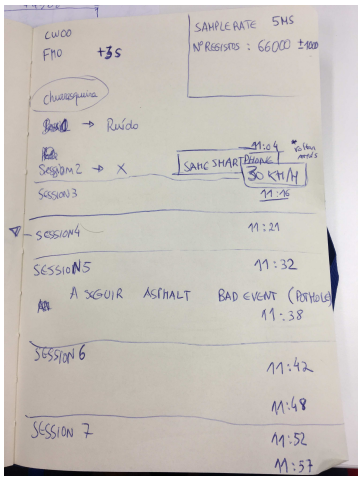
Figure 10 shows notes taken before (Figures 10a and 10c) and during (Figure 10b) this run, which were identified as essential during the aforementioned dry-run. Such notes aided navigating along the itinerary's road anomalies, tracked which sessions were valid, and were later used to process recorded data.

Data from early recording sessions was discarded as they were considered as being part of a warm-up stage. A couple of middle sessions were also disregarded for various reasons, e.g., trucks blocking sections of road with a scheduled anomaly.

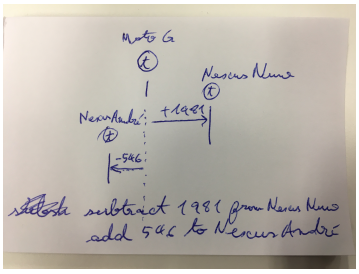
The field study first run was deemed as concluded after successfully finishing five sessions. Finally, it was possible to confirm that all of the recorded sensors data was both available at the smartphone and at the endpoint server.



(a) List of session events expected



(b) Notes on start and end times of sessions and issues faced



(c) Synchronization notes

Figure 10: Notes taken during the first run

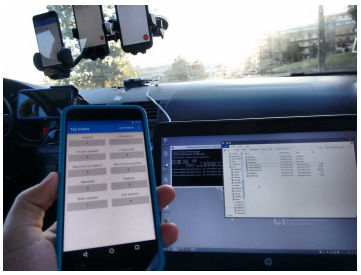
4.2 SECOND RUN

The second run scrutinized data coming from two different smartphones with different sensors. A Nexus 5X and a Samsung Galaxy S Duos were part of the hardware configuration. These smartphones encase a Bosch BMI160 and an Invensense MPU-6000, respectively, to measure acceleration.

Aside from that, there were no other changes to the setup: two iOttie Easy One Touch 3 held both smartphones in identical positions and angles, the recording application sampled the accelerometers at 200 Hz, and the 2007 Mazda 3 was used to travel along the chosen itinerary.

In order to save time and other resources, the hardware configuration was adjusted so multiple field studies could take place at the same time – with due diligence to ensure the validity of all of them. Thus, second and third runs were concurrently performed, as an examination of Figure 11 can reveal.

There was a noticeable drop in recording sessions invalidated by mistakes originated in the team of researchers; but Murphy's law always takes its toll and, due to external factors, some tries were needed to successfully complete five valid sessions. As in previous times, sensors data collected during the warm-up sessions was not analyzed, albeit having been properly stored.



(a) One instance of the annotations application and three instances of the recording application running on multiple mobile devices

(b) Using combinations of three smartphones and three car mounts allowed to concurrently execute two experiments

(c) Detailed view of the unbranded car mount, 1€ coin for scale

Figure 11: Equipment setup for second and third runs. A Bosch Cross Domain Development Kit (XDK)[2] is coupled to the dashboard and connected to a laptop, but results from that experiment are not discussed in this context since they are out of this dissertation's scope

4.3 THIRD RUN

In the third run, the setup with two different car mounts was tested. One of them was an iOttie Easy One Touch 3 and the other was an unbranded equipment,[3] holding the mobile devices in identical positions and angles.

As for the other items in the setup, they all remain unchanged if compared to the standard setup: two Nexus 5X were running the same recording application sampling the accelerometers at 200 Hz, inside a 2007 Mazda 3.

As disclosed in Section 4.2, this experiment was simultaneously performed with the second run. The only implication of this fact was the setup configuration used, which had to be properly planned. Figure 11 has more details on this.

The previous section already talks about the few difficulties encountered by the researching team. This run was deemed as concluded after finishing five valid sessions. Sensors data collected during warm-up sessions was stored but not evaluated, as in previous circumstances.

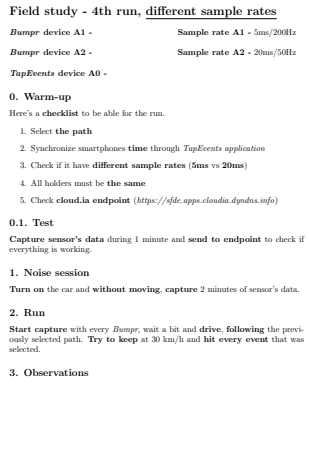
4.4 FOURTH RUN

For the fourth run, two different rates for sampling the sensors data were studied: 200 Hz and 50 Hz (data read each 5 ms and 20 ms, respectively).[4]

2 Details at https://xdk.bosch-connectivity.com/

3 Given the inherent difficulty in uniquely identifying an unbranded part, a detailed view of this car mount can be seen on Figure 11c in an effort to better characterize it

4 Reasons to choose these two specific sample rates were already dissected on Section 3.4, refer to it for a detailed explanation



Field study - 4th run, different sample rates

***Bumpr* device A1 -** **Sample rate A1 -** 5ms/200Hz

***Bumpr* device A2 -** **Sample rate A2 -** 20ms/50Hz

***TapEvents* device A0 -**

0. Warm-up

Here's a **checklist** to be able for the run.

1. Select **the path**
2. Synchronize smartphones **time** through *TapEvents application*
3. Check if it have **different sample rates** (**5ms** vs **20ms**)
4. All holders must be **the same**
5. Check **cloud.ia endpoint** (*https://sfde.apps.cloudia.dyndns.info*)

0.1. Test

Capture sensor's data during 1 minute and **send to endpoint** to check if everything is working.

1. Noise session

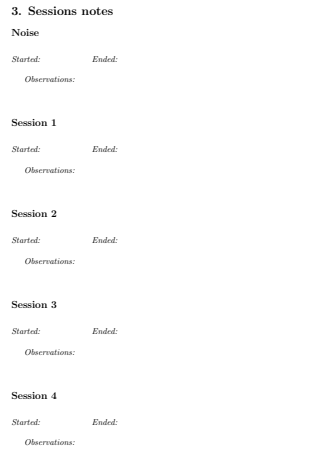
Turn on the car and **without moving**, **capture** 2 minutes of sensor's data.

2. Run

Start capture with every *Bumpr*, wait a bit and **drive**, **following** the previously selected path. **Try to keep** at 30 km/h and **hit every event** that was selected.

3. Observations

(a) A script for the field study, including a checklist



3. Sessions notes

Noise

Started: *Ended:*

Observations:

Session 1

Started: *Ended:*

Observations:

Session 2

Started: *Ended:*

Observations:

Session 3

Started: *Ended:*

Observations:

Session 4

Started: *Ended:*

Observations:

(b) Page to be filled in with sessions' details

Events list - Path 1 (Default University)

The following list shows the sequential events of the selected path.

1. Pothole
2. Pothole
3. Pothole
4. Longbump
5. **Cobblestone**
6. **Asphalt**
7. Manhole
8. Pothole *(University entrance)*
9. Longbump
10. Shortbump
11. Manhole *(University lobby)*
12. Manhole
13. Shortbump
14. Pothole
15. Shortbump
16. Shortbump
17. Shortbump
18. Shortbump
19. Shortbump
20. Shortbump
21. Shortbump

(c) Events to traverse by during the trip

Figure 12: Fourth run notes. Handwritten notes from Figure 10 were used as a template and converted to proper documents

Like in previous runs, all of the other setup parts were kept unchanged. Two iOttie Easy One Touch 3 gripped two identical Nexus 5X running the same recording application (albeit configured to operate with different sampling rates) inside a 2007 Mazda 3.

Likewise to what was accomplished in the second and third runs, fourth and fifth field studies were jointly conducted. This time it was easier to devise a hardware configuration, as two cars were available due to the nature of the fifth run. Thus, Mazda's setup resembled the one already shown in Figure 9, with the only difference being the chosen sampling rate.

After finishing a couple of sessions, one of the researchers noticed something unusual in the recording application's interface – the GPS icon was not featuring in the status bar. Additional inspection confirmed that location updates were not being recorded due to a misconfiguration of the application's permissions.

Making sure the location permission was switched on did not feature in the field study checklist (see Figure 12a), so it was marked as an improvement for eventual future runs. It should be noted, though, that this incident should not have any impact on the validity of sensors data – nevertheless, all data from such sessions was discarded as well as data from warm-up sessions. Upon fixing this issue there were five consecutive valid sessions, which concluded this run.

4.5 FIFTH RUN

Lastly, the fifth run probed two different vehicles, a 2007 Mazda 3 and a 2016 Volkswagen Polo, each driven by a team of researchers.

(a) Both cars in preparation for the experiment

(b) Setup representing the behavior adopted during a session, with the second car tailgating the first. Car's head unit shows a phone call between both vehicles. Speed shown is above $30\,\mathrm{km\,h^{-1}}$ because the photo was taken during a warm-up session

Figure 13: [Identifiable information occluded.] Vehicles and setups used to perform the fifth run

The two cars had a similar setup to those described in previous runs. Both had an iOttie Easy One Touch 3 affixing a Nexus 5X running the recording application at 200 Hz. Figure 13 illustrates this.

Like already acknowledged in Section 4.4, both runs occurred at the same time. This had slightly different implications for the field study if compared with what happened in the second and third runs. For one, the Polo was a rented car and so had no permission to travel inside University of Minho. Because of that, the course had to be adjusted and the portion inside the Campus of Gualtar was switched for a different path with similar length.[5]

Another issue with making an experiment with two different cars is the impossibility to travel the road in the same exact positions, or even at the precisely the same speeds. To cover these problems, the driver of the vehicle in the rear tried to keep a consistent distance to the one in front of it, trying to move like a single unit (see Figure 13b). To assist this task, a car with cruise control was intentionally chosen and teams from both cars were in constant communication via a hands-free phone call.

Problems with the recording application were already described in the previous section. After that, five valid sessions were accomplished in a row and the run was completed. As of always, data acquired during warm-up and invalid sessions was stored but discarded for the purposes of this dissertation.

5 The number and types of road anomalies to be run over was chosen in advance with the intent to mimic the replaced roadway as approximately as possible

RUNS SUMMARY

On a final note, remains to be said that each of the described field studies took about 1 hour to carry out, accounting for all warm-up, invalid, and valid sessions. Usually, there were two warm-up sessions and an average of three invalid sessions for each run. All valid sessions lasted for 6 to 7 minutes.

In total, around 36 hours of driving were monitored, resulting in over 26 million records stored by the backend application. This includes data coming from preliminary validations of the recording application. Furthermore, the number of hours is also a result of having multiple smartphones recording sensors data in parallel in some of the experiments.

5

DATA ANALYSIS

Upon completion of the data collection, it is time to examine it. This chapter documents the data analysis process, beginning with the statistical description of the data, passing through the detection and removal of atypical cases, and finishing by testing the hypotheses formulated on Section 3.2.

Section 5.1 makes use of descriptive statistics to help understanding data's central tendency and dispersion. This information is then used in Section 5.2 to detect incorrect and outlier values and explain the decisions behind the removal, or lack thereof, of subsets of the sensors data. On Section 5.3, the correlation coefficients for the vertical acceleration data are computed to test the previously formulated hypotheses.

5.1 DATA DESCRIPTION

Tables 2 to 4 present descriptive statistics for the runs, broken down by session. In addition to the number of accelerometer observations (samples), the tables show mean ($\bar{x}$), median ($\tilde{x}$), mode, minimum (min), maximum (max), and standard deviation (σ).

Table 2 shows data from the first run, with each horizontal band grouping a successful session, and each of the rows in a band regarding one of the two similar Nexus 5X used. So, both setups A and B had similar configurations as they were used as the control group. Refer to Section 4.1 for additional details.

To improve conciseness, both the second and third runs are represented in Table 3. As already discussed in Chapter 4, these two field studies were executed concurrently, so each horizontal band on the table displays the three different smartphones used in a successful session. Setups C and D correspond to the second run, and setups D and E refer to the third run. Setup D had the standard configuration, setup C had a different smartphone model, and setup E had a different car mount. Sections 4.2 and 4.3 provide better explanations on their differences.

For the same reason, the fourth and fifth runs appear jointly in Table 4. Again, one horizontal band describes a successful session, with a row for each of the three different smartphones utilized. Setups F and G pertain to the fourth run, and setups G and H are related

Table 2: Descriptive statistics for the first run. Each horizontal band groups a successful session. Highlighted row shows incorrect data found after analysis, with all data from the corresponding session being treated as invalid

Setup	samples	$\bar{x}$ ($m\,s^{-2}$)	$\tilde{x}$ ($m\,s^{-2}$)	mode ($m\,s^{-2}$)	min ($m\,s^{-2}$)	max ($m\,s^{-2}$)	σ ($m\,s^{-2}$)
A	66 079	−0.020	−0.018	0.035	−13.255	8.758	1.051
B	3047	0.008	−0.019	−0.010	−0.393	0.388	0.146
A	65 078	−0.031	−0.032	0.009	−16.833	9.010	1.064
B	65 234	−0.021	−0.027	0.047	−19.159	15.892	1.163
A	67 326	−0.021	−0.019	0.065	−16.655	8.721	1.046
B	67 322	−0.001	−0.017	0.120	−21.392	15.189	1.098
A	65 928	−0.029	−0.022	0.045	−19.230	9.884	1.042
B	65 648	−0.012	−0.032	−0.080	−23.336	16.941	1.108
A	66 384	−0.034	−0.034	−0.055	−18.669	10.797	1.041
B	66 386	−0.029	−0.042	−0.075	−21.990	15.600	1.109

to the fifth run. Setup G had the standard configuration already described, setup F had a different sample rate, and setup H was in a different vehicle. For further information on these variations, see Sections 4.4 and 4.5.

From such tables, it is possible to realize that most of the data points are clustered around $0\,m\,s^{-2}$, with a standard deviation of about $1\,m\,s^{-2}$. This falls in line with the expectations, as usually the vehicle is not accelerating in the vertical axis, apart from those brief moments when a road anomaly comes across.[1]

The median value is consistently close to the mean, indicating that values are fairly distributed on the left and right side of the average value. It also signals there being no outliers skewing the dataset – or, at least, it signals that such outliers exist with approximately equal frequency on both sides of the median value.

Despite the relatively small standard deviation, minimum and maximum values are quite afar from the central points, yielding a high range. Points with values so farther apart are associated with the road anomalies which cause the vehicle to rapidly move on the vertical axis, provoking a spike in the monitored acceleration.[2] Despite looking like outliers, these data points increase SNR in the datasets and should not be discarded.

1 Granted, some vertical acceleration is due to fluctuations in a road's slope, but its rate of change is rarely big enough to have a significant impact on the measurements

2 Generally, a single anomaly generates at least two noticeable spikes, one in the positive and then another in the negative direction – or vice versa, depending on the type of anomaly. It is also common that such spikes continue to be observed for some time while the vehicle keeps vibrating in the vertical axis as a result of the impact

Table 3: Descriptive statistics for the second and third runs. Each horizontal band groups a successful session. Setups C and D correspond to the second run, and setups D and E refer to the third run

Setup	samples	$\bar{x}$ ($m\,s^{-2}$)	$\tilde{x}$ ($m\,s^{-2}$)	mode ($m\,s^{-2}$)	min ($m\,s^{-2}$)	max ($m\,s^{-2}$)	σ ($m\,s^{-2}$)
C	77 593	−0.020	−0.016	0.019	−12.318	8.077	1.149
D	78 013	0.027	0.017	0.147	−9.994	8.084	0.969
E	77 955	0.004	0.016	0.174	−8.745	10.274	0.946
C	76 696	−0.023	−0.027	−0.085	−11.607	7.836	1.079
D	76 671	−0.026	−0.041	−0.036	−9.252	8.155	0.944
E	76 577	0.006	0.006	0.075	−9.773	8.764	0.959
C	75 133	−0.011	−0.012	0.116	−11.676	9.883	1.132
D	74 491	−0.015	−0.031	−0.059	−7.539	7.186	0.932
E	74 760	−0.021	−0.025	−0.181	−9.854	9.198	0.967
C	75 041	−0.020	−0.022	−0.027	−10.489	8.567	1.092
D	73 987	−0.008	−0.023	0.310	−7.879	7.006	0.932
E	74 625	−0.019	−0.020	0.085	−9.608	7.876	0.960
C	86 266	−0.018	−0.020	0.002	−12.487	7.817	0.986
D	85 832	−0.017	−0.031	0.053	−10.075	8.297	0.875
E	85 893	−0.010	−0.008	−0.015	−10.182	9.713	0.907

Table 4: Descriptive statistics for the fourth and fifth runs. Each horizontal band groups a successful session. Setups F and G pertain to the fourth run, and setups G and H are related to the fifth run

Setup	samples	$\bar{x}$ ($m\,s^{-2}$)	$\tilde{x}$ ($m\,s^{-2}$)	mode ($m\,s^{-2}$)	min ($m\,s^{-2}$)	max ($m\,s^{-2}$)	σ ($m\,s^{-2}$)
F	19 424	0.013	0.008	−0.056	−6.492	6.149	0.816
G	77 796	0.012	0.003	−0.059	−11.716	7.257	0.894
H	78 573	0.019	0.021	0.090	−7.301	7.429	0.796
F	17 070	0.011	0.006	−0.025	−7.057	6.418	0.864
G	68 207	−0.018	−0.027	−0.173	−8.786	8.106	0.918
H	68 666	0.003	0.004	−0.023	−7.756	6.121	0.836
F	18 491	0.011	0.000	−0.096	−6.876	6.334	0.840
G	74 059	0.013	0.001	−0.059	−10.478	7.262	0.918
H	74 397	0.005	0.007	−0.113	−7.075	7.786	0.833
F	19 844	0.007	0.000	−0.206	−5.748	5.489	0.798
G	79 330	0.018	0.013	−0.020	−8.419	6.945	0.847
H	78 906	−0.014	−0.018	−0.023	−7.474	6.990	0.807
F	16 159	0.018	0.016	−0.010	−5.679	6.360	0.899
G	64 709	−0.001	−0.007	−0.060	−8.378	7.209	0.973
H	65 128	0.006	0.009	−0.003	−6.639	7.436	0.889

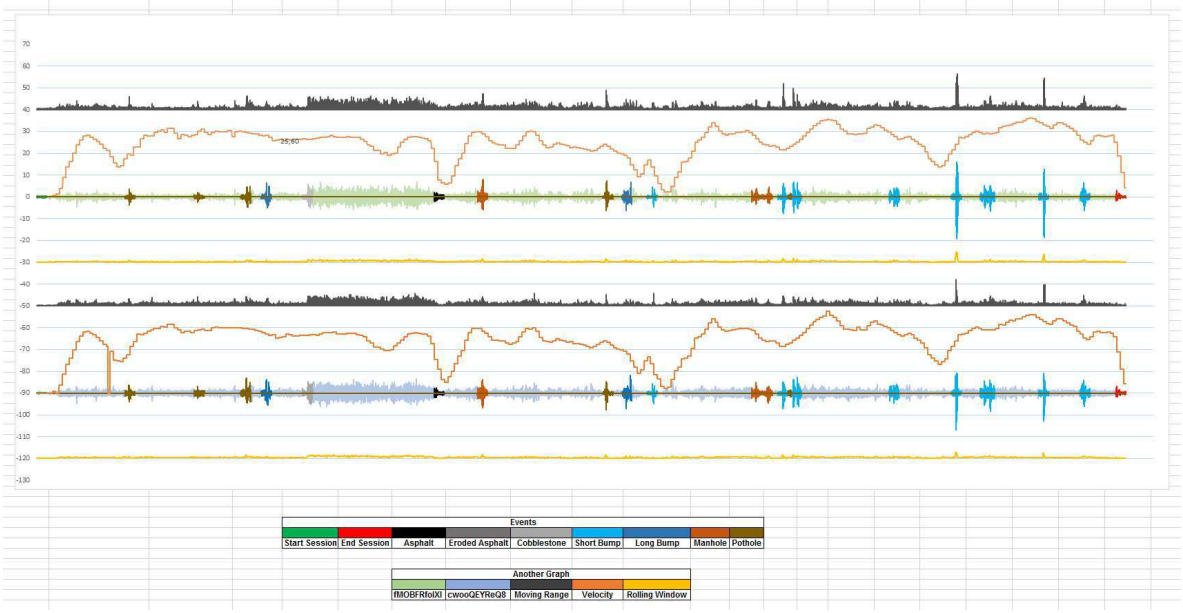


Figure 14: [Best viewed in color.] Data spreadsheet summarizing a session from the first run, including annotations of the road anomalies. Vertical acceleration ($\mathrm{m\,s^{-2}}$) for one of the smartphones is plotted in light green (around the vertical value of 0); the other smartphone's vertical acceleration is charted in grayish blue (around the vertical value of -90, shifted to improve readability). Orange lines show the speed reported by each smartphone in $\mathrm{km\,h^{-1}}$

5.2 DATA SET REDUCTION

In addition to the analysis of statistical data, data validation was performed by means of visual inspection using a data spreadsheet – Figure 14 shows an example. This tool was developed to organize information in such way that multiple smoke tests could be performed with it.

For one, it was possible to assert that every single road anomaly had been correctly identified using the annotations application. Since the anomalies were represented as an interval of 3 seconds, it was verifiable that expected acceleration spikes occurred within it. Furthermore, the continuous line of the vertical acceleration demonstrated that no data was missing. Vertically aligning measurements from both smartphones allowed to confirm their data was properly synchronized.

Since sensors data started to be recorded before the vehicle initiated the trip and stopped to be recorded after the vehicle terminated it, all datasets included some data corresponding to periods where the vehicle was stationary. In those periods, smartphone sensors were essentially monitoring the vibrations caused by the engine, fluctuating around $0\,\mathrm{m\,s^{-2}}$, in what can be categorized as noise.

Thus, as a way to improve the SNR of the datasets, initial and final sections of the data were removed. Making use of data collected during the sessions with the annotations ap-

plication, sensors data prior to the start of a session was clipped; the same method was performed for sensors data after the end of a session.

An analysis of Table 2 detected incorrect data in the first session, with one of the smartphones reporting a very small number of observations (see highlighted row). Additional inspection concluded that such data was missing and could not be recovered, so all data belonging to first run's first session was treated as invalid.

Finally, it lasts to be noted that discrepancies in the count of samples observed in Table 4 are expected, as this table includes a run where different sample rates were tested. One of the smartphones reports 4 times fewer readings than the order two, which is consistent with the relationship between the chosen sample rates – 200 Hz versus 50 Hz.

5.3 HYPOTHESIS TESTING

The four hypotheses formulated in Section 3.2 were tested with the techniques presented in Section 3.7. A specially designed tool described in Section 3.8.3, TimeWarper, was used to assist in this effort.

To briefly recap what was discussed in Chapter 3, this application takes two vertical acceleration datasets, computes the warping path between them using a DTW algorithm, warps the datasets, and then computes the correlation coefficient. The correlation coefficient is a normalized value, varying between -1 (inversely correlated) and 1 (perfectly correlated); a value of 0 means the time series are entirely uncorrelated. After this, a statistical validation is performed and the results are logged to a file.

Data collected in the control experiment sets the baseline correlation coefficient to which the other coefficients will be compared. These comparisons will allow making decisions about the proposed hypotheses.

Table 5: Correlation coefficients by run and session. Highlighted cell shows a session for which it was not possible to compute the correlation coefficient due to invalid data, corresponding to the highlighted row in Table 2

Session	Run				
	1st (baseline)	2nd (smartphone)	3rd (car mount)	4th (samp. rate)	5th (car)
1	—	0.834	0.845	0.841	0.826
2	0.889	0.828	0.843	0.831	0.835
3	0.892	0.826	0.855	0.836	0.822
4	0.892	0.831	0.852	0.829	0.825
5	0.894	0.830	0.846	0.833	0.825
$\bar{x}$	0.892	0.830	0.848	0.834	0.827

Table 5 shows the computed coefficients for all valid sessions on every run, along with the mean value ($\bar{x}$). Despite only making use of these mean values to illustrate the following arguments, every individual coefficient was statistically validated. For reasons discussed in the previous section, the first run's first session was treated as invalid, so the correlation coefficient was not computed. The mean value for the first run is computed over the remaining four valid values.

There is an expectation for the correlation coefficient to be very high for two similar collection setups sensing the vertical acceleration during a recording session. The control experiment, detailed in Section 4.1, allows testing this expectation.

Running all the first run's valid sessions trough the TimeWarper tool yields a mean correlation coefficient of 0.892 (see Table 5). According to the classification proposed by Evans [67], this coefficient represents a strong positive correlation.

To test the statistical significance of this result, each valid session was processed using the following technique. Let us start by assuming that the result has no significance. If so, it follows that computing the correlation of data with nothing but noise would produce similar correlation coefficients. One can produce "noised" versions of the same data by rearranging the order of its data points.

Using an RS algorithm, 100 randomized copies of each smartphone's vertical acceleration data were produced – these copies are called surrogates. Then, each pair of surrogates is warped and its correlation coefficient is computed. Lastly, the coefficients are rank ordered.

The original assumption can be rejected if the correlation coefficient for the original pair, r_0, is at the tails of the coefficients distribution. For a significance level of $\alpha = 0.05$, if the rank of r_0 in the ordered list of coefficients:

$$\text{is less than } (100+1)\,\frac{0.05}{2} \approx 3 \text{ or is greater than } (100+1)\left(1-\frac{0.05}{2}\right) \approx 98$$

then the assumption is rejected and the result is statistically significant.

Figure 16 plots the ordered lists of coefficients for the first run.[3] For all sessions, the original correlation is placed at the tail of each list, ranking at the 101^{st} position which is greater than what is required. This is very far from all surrogate pairs[4] – in fact, a better significance level of $\alpha = 0.01$ would lead to the same conclusions.

The initial result of 0.892 is thus considered as valid and will be used as the baseline for the experiments analyzed below.

3 To improve this section's readability, the charts showing the correlation coefficients were moved to Appendix D, on pages 82 through 86

4 The mean value for the correlation coefficients of the pairs of surrogates is consistently bellow 0.8, the threshold for strong positive correlation [67]. These results show that the bias introduced by DTW in the correlation coefficient is not large enough to incorrectly classify noisy data as being strongly correlated

In contrast with the control experiment, there is an expectation that changing the independent variables will yield a smaller correlation coefficient than the baseline. However, it is difficult to have an intuition for the magnitude of this difference.

Initially formulated on Section 3.2, Hypothesis 1_0 states that there is no similarity in the measurements of vertical acceleration when using different smartphones. To test it, data from the second run was fed into TimeWarper, which resulted in a mean correlation coefficient of 0.830 (see Table 5), also a strong positive correlation [67].

The statistical significance test followed the same procedure as described above: start by assuming no significance, produce surrogate pairs, warp and correlate them, reject the initial assumption if the original correlation is at the tails of the ordered list. Figure 17 plots this ordered list and, again, the original coefficient is very far from the surrogate pairs' coefficients in every session (also ranking at 101^{st}), validating the result of 0.830.

This coefficient shows a strong positive correlation between measurements of vertical acceleration when using different smartphones, which refutes the null hypothesis. Thus, Hypothesis 1_1, the alternative hypothesis stating that such measurements are similar, must be true.

Hypothesis 2_0 declares that there is no similarity in the measurements of vertical acceleration when using different car mounts. Analysis of data from the third run using the TimeWarper tool returned a mean correlation coefficient of 0.848 (see Table 5), a strong positive correlation too [67].

The same procedure as described before was put in place to test the statistical significance of this result. Figure 18 shows the ordered list which validates the correlation coefficient (once again, at the 101^{st} position).

As before, the resulting strong positive correlation between measurements of vertical acceleration when using different car mounts refutes the null hypothesis. Therefore, Hypothesis 2_1, the alternative hypothesis, must be true.

Hypothesis 3_0 affirms that there is no similarity in the measurements of vertical acceleration when using different sampling rates. TimeWarper analyzed data from the fourth run and the result has shown a mean correlation coefficient of 0.834 (see Table 5), again a strong positive correlation [67].

This result's statistical significance was tested in the same manner as described above. The correlation coefficient was validated by the ordered list presented in Figure 19 (original pair ranked 101^{st}, too).

Again, the result is a strong positive correlation between measurements of vertical acceleration when sampling the sensors at different rates, which disproves the null hypothesis. This implies a value of truth for the alternative hypothesis, Hypothesis 3_1.

Finally, Hypothesis 4_0 asserts that there is no similarity in the measurements of vertical acceleration when using different vehicles. Data collected during the fifth run was examined using the TimeWarper tool, outputting a mean correlation coefficient of 0.827 (see Table 5), once more a strong positive correlation [67].

Like in the previous paragraphs, the same procedure was conducted to statistically validate the results. The ordered list displayed on Figure 20 validated the correlation coefficient (yet again, original pair appearing as 101^{st}).

With strong positive correlation between measurements of vertical acceleration when driving different vehicles, the null hypothesis is also rejected. So the alternative hypothesis, Hypothesis 4_1, must be true.

6

DISCUSSION OF RESULTS

This chapter discusses the results achieved during the experimental work. Threats to the experiments' validity are examined along with inferences on the results effectiveness for the population of each studied variable. Finally, lessons learned during the experiment are stated in order to improve the reproducibility of this study.

6.1 RESULTS INTERPRETATION

Presented in the previous chapter, Table 5 summarizes the results obtained in this study. All independent variables have a similar impact on the computed coefficients, except for the car mount which has a smaller effect. Granted, these variations could be due to the small sample size, but that is not very likely given the low standard deviation of the coefficients computed for each run.

At the very least, these results show that smartphones are adequate for prototyping purposes during the development of ADAS in the context of vertical acceleration. They also indicate that information collected from smartphone sensors may be robust enough for the potential development of such systems.

In practical terms, all results point to the same core idea: changing one of the studied variables in the setup does not have a major impact on the vertical acceleration collection process. This has strong implications for the development of ADAS using smartphone sensors, since it opens the possibility to develop new systems in a cost-effective approach, e.g., by using cheaper smartphones or simply by making use of smartphones from the vehicles' drivers.

For instance, suppose a system is developed to determine the condition of a road based on vertical acceleration as sensed by a particular smartphone, namely a Nexus 5X. Results obtained in this document indicate that a Samsung Galaxy S Duos could also be used as sensing device without affecting the quality of information acquired. Likewise, it follows that using other similar smartphones would result in an adequate performance.

A similar argument could be made for the vehicle, the rate of sampling, or the car mount. Any car similar to the ones used during the fifth run should yield similar data, allowing for the deployment of an ADAS to a large section of the VIO. A smaller rate of sampling would be good enough, allowing for better use of resources like the battery or mobile data.

Likewise, since the two car mounts studied were very different in what concerns to the build quality, it is fairly accurate to state that any car mount would be suitable to use in such system. In the context of what was discussed in Section 3.1 about avoiding the use of car mounts, these results give some confidence to the possibility of developing a system where the smartphone is fixed inside the driver's pocket.

6.2 THREATS IDENTIFICATION

It is possible that some decisions made during the experiment planning had an impact on the validity of the results obtained. The following overview identifies those potential threats to validity – and measures put in place to address them – in order to document opportunities for further research.

The sample size – five repetitions of each experimental study – is relatively small. Unfortunately, constraints on time and other resources dictated this and were not possible to avoid.

One reassuring fact regarding this concern is the small standard deviation found while computing the coefficient correlations, which brings more statistical confidence to the results. The randomization tests performed also prove that, at the very least, the results are not attributable to mere noise.

Fourth and fifth runs were performed in an itinerary slightly different from the one previously planned. This was due to the usage of a rented car without permission to travel inside the Campus of Gualtar.

To counter this issue, a new itinerary was planned with an approximate number of road anomalies and types of anomaly. It is possible for this change of plans to have an impact on the correlation coefficients computed for these two runs, making it difficult to compare those values against the baseline set by the first run. Nevertheless, the mean correlation coefficients are in line with the values reported in other experiments, giving some confidence in these results.

Despite being an adequate data processing methodology, with several prior research backing it, DTW introduces bias in the computed correlation coefficients. Therefore, relying on DTW may pose a threat to the relevance of this study since it is difficult to directly compare its results against the coefficients obtained with other methods.

The speed used by the vehicles to travel along the itinerary influences the vertical acceleration measured by the smartphones. For this reason, a test considering the vehicles' speed as an independent variable would be relevant but was not possible due to the context of the experiments – city roads with normal traffic conditions. To achieve this goal, a test track and cars equipped with cruise control would be needed.

As a measure to mitigate the effect of not having the speed as a controlled variable, an arbitrary value was chosen for the experiments, with the vehicle's driving being responsible for maintaining it. The speed was also monitored during the experiments and used during the data analysis phase to decide on the validity of each recording session.

Two very similar cars were used to represent the vehicles; while being true that both of them represent – in a very broad sense – typical cars, they might not be different enough to represent a wide variety of VIO, like buses, trucks, or motorcycles. An extensive analysis of this topic required a level of control not achievable in the scope of this dissertation.

This consideration can be generalized to all subjects chosen to represent the independent variables. Despite the effort in choosing two rather different subjects for each category, some doubts remain about the effects of using a broader range of, e.g., smartphone sensors or car mounts.

For that purpose, a more detailed study could have been performed for each variable. However, the context of this dissertation focused on a breadth-first approach instead of a depth-first one: it was deemed as more important to acquire some knowledge about many independent variables, rather than knowing a lot about only one of them.

6.3 INFERENCING

Taking into account the threats listed in the previous chapter, inferences can be made on the results for each studied variable's population.

In the case of smartphones and their inertial sensors, both tested devices had IMUs of similar capabilities from two different manufacturers. The inertial sensors embedded in smartphones tend to output data with "good enough" quality, so similar results to this dissertation are expected for a wide variety of smartphones.

This expectation's degree of confidence is higher for smartphones with similar or higher price points as the Nexus 5X and the Samsung Galaxy S Duos, but additional tests with a Wiko Sunny – available around the 50€ mark – have shown this device to also have adequate capabilities.

There is an indication that virtually any car mount capable of holding a smartphone to the windshield can be used while collecting smartphone sensors data. This is due to the quality

of the samples used during the experiment – a very good car mount versus a low-quality one. Thus, it is expected that other car mounts inside this range to have an adequate performance.

As to the rate of sampling, anything between 200 Hz to 50 Hz seems to be sufficient. While deciding on which rate to choose, the following tradeoff should be considered: is the small gain in quality of information worth the four-fold increase in data to be recorded and processed? The answer, depending on the use case, should generally be negative. This is in line with previous research which states that 50 Hz is adequate for the context of smartphone sensing.

Currently, increasing the rate of sampling above 200 Hz is impractical. The technology for higher rates certainly exists, but the rise in quality does not seem to be cost-effective for manufacturers since the value delivered to smartphones customers does not increase at the same pace as the price for higher-end inertial sensors.[1] Decreasing the rate of sampling below 50 Hz has a significant impact on the information collected, as previous research as shown. Below this threshold, the previously presented tradeoff is actually reversed: is a decrease in the amount of data recorded worth the sharp drop in quality of information?

As previously discussed, it is difficult to estimate the results for a broad range of vehicles. It is not clear if the same conclusions of this dissertation will hold, e.g., for buses where the level of vibration due to the motor can significantly increase the noise in measurements. In this area, more research is needed.

6.4 IDENTIFICATION OF LEARNED LESSONS

To aid researchers trying to replicate the experiments described in this dissertation – in addition to the predictable advice to have a proper planning phase – it is recommended a good level of preparation for responding to changes. Small adjustments are unavoidable in an experiment of this nature, in city roads. Because of this, it is better to invest some time in planning how to deal with changes rather than thoroughly detailing all the steps and then fail to react when things inevitably go south.

It is recommended to perform the experiments with a team of no less than three researchers: one for driving and maintaining the vehicle's speed, another to annotate the road anomalies found during the trip, and yet another for aiding in navigation during the trip. Doing otherwise was found impractical during the dry-run experiment and consequently adjusted for the subsequent experiments.

1 This is related to the law of diminishing marginal utility. In economics, marginal utility measures the change in satisfaction from increasing the consumption of a good or service – in this case, adapted to the increase in quality. For instance, to a smartphone customer, a smartphone with any kind of accelerometer has a very high marginal utility since it allows for features such as automatic screen rotation. Having a slightly better accelerometer has a lower – but still positive – marginal utility, enabling new functionalities like an always-on pedometer. However, having a very high-end accelerometer does not seem to add much utility as it does not bring any additional features – in fact, it might even have negative marginal utility due to a potential increase in battery consumption

Finally, it is essential to adapt existing tools – or to develop adequate ones – for using when analyzing the acquired data. This can seem as a time sink in early phases, but the investment in automation will pay off as the study approaches its end. Trying to elicit the requirements for such tools can even guide and clarify the experiment planning.

7

CONCLUSIONS

To wrap up all the information presented in this document, final considerations are stated along with suggestions for future work motivated by the conclusions.

7.1 FINAL CONSIDERATIONS

This dissertation's main contribution is an experimental study on the impact in the quality of data collected by smartphones when using different smartphones, car mounts, rates of sampling, or vehicles for the purpose of ADAS development. This study shows that the quality of data acquired with smartphone sensors is not significantly affected by using different variations of those elements. These results indicate that it may be feasible to use smartphone sensors for the prototyping and development of ADAS without the need to standardize the components used.

Another contribution is the data acquisition application based on a smartphone, which is able to collect and store smartphone sensors data and subsequently sending such data to an endpoint where it can be further analyzed.

Furthermore, the source code of a tool to automate the data analysis is provided, easing the effort of trying to replicate this study or allowing for similar studies to be performed with less overhead.

7.2 FUTURE WORK

Experimental results documented in this dissertation can motivate further investigation in similar contexts. As identified in Section 6.2, more thorough studies can be performed for any of the independent variables to strengthen the confidence of the results. Preferably, such studies should both have a greater number of repetitions and should study a wider variety of subjects, e.g., by testing different types of vehicles.

In particular, it would be interesting to see a further investigation on the car mounts, as their higher mean correlation coefficient seems to be counterintuitive. A possible explanation of

such good results might be related to the extra movement of the car mounts being canceled by the vehicle's vibrations. The opposite could also be true: the car mounts might amplify the sensed vertical acceleration, leading to higher deltas between the higher and smaller values, resulting in higher correlation coefficients.

As previously identified, a study performed in a context where the vehicles' speed could be controlled and treated as an independent variable would be very valuable. To do so, a test track and cruise control-equipped cars should suffice.

This experiment could be reproduced with a focus in gyroscope data as the dependent variable without introducing major changes. With this intent in mind, gyroscope data was also collected during the vertical acceleration experiments. Such replication will be performed if an eventual use case for such information arises.

A comparison of the capabilities of smartphones versus those provided by special-purpose sensor boxes, like the XDK, is also in perspective. During the second and third runs, one of such devices was put inside the vehicle in order to collect sensors data for future analysis.

In the context of the innovation program INNOVCAR in which this dissertation was carried out, a system is being developed in partnership with Transportes Urbanos de Braga consisting of a fleet of buses equipped with smartphones. Such smartphones have installed a newer version of the Android application, described in Section 3.8.1, which will gather data from the smartphone sensors. Data will then be sent to a cloud infrastructure, where it will undergo some processing with data mining algorithms. Finally, relevant extracted features of such data will be presented in a browser with the help of a front-end application.

An article based on the experimental results achieved during this dissertation is being prepared. The following conferences were identified as candidates for submission: IEEE Intelligent Transportation Systems Conference (Q3 2018), IEEE Intelligent Vehicles Symposium (Q2 2018), IEEE International Conference on Pervasive Computing and Communications (Q1 2018), and International Conference on Connected Vehicles (Q1 2018).

REFERENCE LIST

[1] Miguel Goulão and Fernando Brito e Abreu. Modeling the Experimental Software Engineering Process. *6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007)*, pages 77–90, 2007. DOI 10.1109/QUATIC.2007.18.

[2] Jonathan W. Schooler. Metascience could rescue the 'replication crisis'. *Nature*, 515 (7525):9–9, nov 2014. DOI 10.1038/515009a.

[3] Jeffrey S. Mogil and Malcolm R. Macleod. No publication without confirmation. *Nature*, 542(7642):409–411, feb 2017. DOI 10.1038/542409a.

[4] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, may 2016. DOI 10.1038/533452a.

[5] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: An Introduction*, volume 6. Springer US, 2000. DOI 10.1007/978-1-4615-4625-2.

[6] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8):721–734, aug 2002. DOI 10.1109/tse.2002.1027796.

[7] Anders Lindgren and Fang Chen. State of the art analysis: An overview of advanced driver assistance systems (ADAS) and possible human factors issues. *Human factors and economics aspects on safety*, pages 38–50, 2006.

[8] Richard Bishop. *Intelligent vehicle technology and trends*. Artech House Publishers, 2005. ISBN 1580539114.

[9] Lie Guo, Xiao-Hui Huang, Ping-Shu Ge, Guang-Xi Zhang, and Ming Yue. Lane changing trajectory tracking control for intelligent vehicle on curved road based on backstepping. *Journal of Jilin University (Engineering and Technology Edition)*, 43(2):323–328, 2013.

[10] Jin Xu, Guang Chen, and Ming Xie. Vision-guided automatic parking for smart car. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 725–730, 2000.

[11] Jie Sun, Zhao-hui Wu, and Gang Pan. Context-aware smart car: from model to prototype. *Journal of Zhejiang University SCIENCE A*, 10(7):1049–1059, 2009. ISSN 1862-1775. DOI 10.1631/jzus.A0820154.

[12] Jeremy Broughton and Chris Baughan. The effectiveness of antilock braking systems in reducing accidents in Great Britain. *Accident Analysis & Prevention*, 34(3):347 – 355, 2002. ISSN 0001-4575. DOI 10.1016/S0001-4575(01)00032-X.

[13] Anders Lie, Claes Tingvall, Maria Krafft, and Anders Kullgren. The effectiveness of ESP (Electronic Stability Program) in reducing real life accidents. March 2004. DOI 10.1080/15389580490269164.

[14] Jaeseok Kim and Hyunchul Shin. Introduction. In Jaeseok Kim and Hyunchul Shin, editors, *Algorithm & SoC Design for Automotive Vision Systems*, pages 1–10. Springer Netherlands, 2014. ISBN 978-94-017-9074-1. DOI 10.1007/978-94-017-9075-8_1.

[15] Alfred Eckert, Andree Hohm, and Stefan Lueke. An integrated ADAS solution for pedestrian collision avoidance. In *23rd International Technical Conference on the Enhanced Safety of Vehicles, Seoul, Korea*, 2013.

[16] Steven E. Shladover. Automated vehicles for highway operations: Automated highway systems. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 219(1):53–75, 2005.

[17] Texas Instruments. Advanced Driver Assistance (ADAS) solutions guide. Technical report, Texas Instruments, 2015.

[18] Jacob Poushter. Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies. *Pew Research Center*, pages 1–5, 2016. URL http://www.pewglobal.org/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies.

[19] Dongyao Chen, Kyong-tak Cho, Sihui Han, Zhizhuo Jin, and Kang G Shin. Invisible Sensing of Vehicle Steering with Smartphones. *Mobile Systems , Application and Services*, pages 1–13, 2015. ISSN 9781450334945. DOI 10.1145/2742647.2742659.

[20] Mohamed Fazeen, Brandon Gozick, Ram Dantu, Moiz Bhukhiya, and Marta C. González. Safe Driving Using Mobile Phones. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1462–1468, 2012. ISSN 15249050. DOI 10.1109/TITS.2012.2187640.

[21] Jarret Engelbrecht, Marthinus Johannes Booysen, Frederick Johannes Bruwer, and Gert-Jan van Rooyen. Survey of smartphone-based sensing in vehicles for intelligent

transportation system applications. *IET Intelligent Transport Systems*, 9(10):924–935, 2015. DOI 10.1049/iet-its.2014.0248.

[22] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones. *Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys '08*, page 323, 2008. DOI 10.1145/1460412.1460444.

[23] Artis Mednis, Girts Strazdins, Reinholds Zviedris, Georgijs Kanonirs, and Leo Selavo. Real time pothole detection using Android smartphones with accelerometers. *2011 International Conference on Distributed Computing in Sensor Systems and Workshops, DCOSS'11*, 2011. DOI 10.1109/DCOSS.2011.5982206.

[24] Vittorio Astarita, Maria Vittoria Caruso, Guido Danieli, Demetrio Carmine Festa, Vincenzo Pasquale Giofrè, Teresa Iuele, and Rosolino Vaiana. A Mobile Application for Road Surface Quality Control: UNIquALroad. *Procedia - Social and Behavioral Sciences*, 54:1135–1144, 2012. DOI 10.1016/j.sbspro.2012.09.828.

[25] Javier Almazan, Luis M. Bergasa, J. Javier Yebes, Rafael Barea, and Roberto Arroyo. Full auto-calibration of a smartphone on board a vehicle using IMU and GPS embedded sensors. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):1374–1380, 2013. ISSN 1931-0587. DOI 10.1109/IVS.2013.6629658.

[26] Marco D. Tundo, Edward Lemaire, and Natalie Baddour. Correcting Smartphone orientation for accelerometer-based analysis. *MeMeA 2013 - IEEE International Symposium on Medical Measurements and Applications, Proceedings*, pages 58–62, 2013. DOI 10.1109/MeMeA.2013.6549706.

[27] Haofu Han, Jiadi Yu, Hongzi Zhu, Yingying Chen, Jie Yang, Yanmin Zhu, Guangtao Xue, and Minglu Li. SenSpeed: Sensing driving conditions to estimate vehicle speed in urban environments. *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 15(JANUARY 2016):727–735, 2014. DOI 10.1109/INFOCOM.2014.6847999.

[28] Marco Piras, Andrea Lingua, Paolo Dabove, and Irene Aicardi. Indoor Navigation Using Smartphone Technology: A future challenge or an actual possibility? *Record - IEEE PLANS, Position Location and Navigation Symposium*, pages 1343–1352, 2014. DOI 10.1109/PLANS.2014.6851509.

[29] Astarita Vittorio, Vaiana Rosolino, Iuele Teresa, Caruso Maria Vittoria, P. Giofrè Vincenzo, and De Masi Francesco. Automated Sensing System for Monitoring of Road Surface Quality by Mobile Devices. *Procedia - Social and Behavioral Sciences*, 111: 242–251, 2 2014. DOI 10.1016/j.sbspro.2014.01.057.

[30] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pages 29–39, 2008. DOI 10.1145/1378600.1378605.

[31] Chih-wei Yi, Yi-ta Chuang, and Chia-sheng Nian. Toward Crowdsourcing-Based Road Pavement Monitoring by Mobile Sensing Technologies. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1905–1917, 8 2015. DOI 10.1109/TITS.2014.2378511.

[32] Yu-chin Tai, Cheng-wei Chan, and Jane Yung-jen Hsu. Automatic road anomaly detection using smart mobile device. *2010 15th Conference on Artificial Intelligence and Applications (TAAI)*, pages 1–8, 2010. URL http://w.csie.org/~yctai/papers/taai2010_paper.pdf.

[33] Mikko Perttunen, Oleksiy Mazhelis, Fengyu Cong, Tapani Ristaniemi, and Jukka Riekki. Distributed Road Surface Condition Monitoring. *Lecture Notes in Computer Science*, pages 64–78, 2011.

[34] Fatjon Seraj, Berend Jan Van Der Zwaag, Arta Dilo, and Tamara Luarasi. RoADS : A road pavement monitoring system for anomaly detection using smart phones. 2014. DOI 10.1007/978-3-319-29009-6_7.

[35] Kun Li, Man Lu, Fenglong Lu, Qin Lv, Li Shang, and Dragan Maksimovic. Personalized driving behavior monitoring and analysis for emerging hybrid vehicles. *Pervasive Computing*, pages 1–19, 2012. DOI 10.1007/978-3-642-31205-2_1.

[36] Johannes Paefgen, Flavius Kehr, Yudan Zhai, and Florian Michahelles. Driving behavior analysis with smartphones: insights from a controlled field study. *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, (January): 36:1–36:8, 2012. DOI 10.1145/2406367.2406412.

[37] Yuchuan Du, Chenglong Liu, Difei Wu, and Shengchuan Jiang. Measurement of International Roughness Index by using Z-axis accelerometers and GPS. *Mathematical Problems in Engineering*, 2014, 2014. ISSN 15635147. DOI 10.1155/2014/928980.

[38] German Castignani, Thierry Derrmann, Raphael Frank, and Thomas Engel. Driver Behavior Profiling Using Smartphones: A Low-Cost Platform for Driver Monitoring. *IEEE Intelligent Transportation Systems Magazine*, 7(1):91–102, 1 2015. DOI 10.1109/MITS.2014.2328673.

[39] D. W. Allan, N. Ashby, and C. C. Hodge. *The science of timekeeping*. Hewlett Packard, 1997. ISBN 1289. URL http://www.allanstime.com/Publications/DWA/Science_Timekeeping/TheScienceOfTimekeeping.pdf.

[40] L. Ruiz-Garcia, P. Barreiro, and J. I. Robla. Performance of ZigBee-Based wireless sensor nodes for real-time monitoring of fruit logistics. *Journal of Food Engineering*, 87 (3):405–415, 2008. ISSN 02608774. DOI 10.1016/j.jfoodeng.2007.12.033.

[41] P. Dabove, G. Ghinamo, and A. M. Lingua. Inertial sensors for smartphones navigation. *SpringerPlus*, 4(1):834, 2015. DOI 10.1186/s40064-015-1572-8.

[42] Kasun De Zoysa, Chamath Keppitiyagama, Gihan P Seneviratne, and W W A T Shihan. A Public Transport System Based Sensor Network for Road Surface Condition Monitoring. *Proceedings of the 2007 Workshop on Networked Systems for Developing Regions*, pages 9:1–9:6, 2007. DOI 10.1145/1326571.1326585.

[43] Kongyang Chen, Mingming Lu, Xiaopeng Fan, Mingming Wei, and Jinwu Wu. Road condition monitoring using on-board three-axis accelerometer and GPS sensor. *Proceedings of the 2011 6th International ICST Conference on Communications and Networking in China, CHINACOM 2011*, (2009):1032–1037, 2011. DOI 10.1109/ChinaCom.2011.6158308.

[44] Artis Mednis, Girts Strazdins, Martins Liepins, Andris Gordjusins, and Leo Selavo. RoadMic: Road surface monitoring using vehicular sensor networks with microphones. *Communications in Computer and Information Science*, 88 CCIS(PART 2):417–429, 2010. ISSN 18650929. DOI 10.1007/978-3-642-14306-9_42.

[45] Christopher Nowakowski, Somak Datta Gupta, Scott Myers, Steven Shladover, Joe Butler, and Alex Bayen. Providing In-Vehicle Soft Safety Alerts Using Mobile Millennium Data and Vehicle Event Information - Final Report For Renault. Technical report, University of California, California, 2012. URL http://escholarship.org/uc/item/5n50d6rz.

[46] Zoltán Bankó and János Abonyi. Correlation based dynamic time warping of multivariate time series. *Expert Systems with Applications*, 39(17):12814–12823, 2012. ISSN 09574174. DOI 10.1016/j.eswa.2012.05.012.

[47] Jim Scarlett. Enhancing the Performance of Pedometers Using a Single Accelerometer. *Analog Device*, pages 1–16, 2007. URL http://www.analog.com/library/analogDialogue/archives/41-03/pedometer.html.

[48] H. Eren, S. Makinist, E. Akin, and A. Yilmaz. Estimating driving behavior by a smartphone. *IEEE Intelligent Vehicles Symposium, Proceedings*, (254):234–239, 2012. DOI 10.1109/IVS.2012.6232298.

[49] Radoslav Stoichkov. Android Smartphone Application for Driving Style Recognition. *Lehrstuhl fur Medientechnik, Technische*, 2013. URL http://www.

eislab.fim.uni-passau.de/files/publications/students/Stoichkov-Projektarbeit.pdf.

[50] Meinard Müller. Dynamic Time Warping. In *Information retrieval for music and motion (2007): 69-84*, chapter 4, page 318. Springer-Verlag Berlin Heidelberg, 2007. ISBN 978-3-540-74047-6. DOI 10.1007/978-3-540-74048-3.

[51] Stan Salvador and Philip Chan. FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intelligent Data Analysis*, 11:561–580, 2007. ISSN 1088467X. URL http://www.cs.fit.edu/~pkc/papers/tdm04.pdf.

[52] EG Caiani, A Porta, G Baselli, M Turiel, S Muzzupappa, F Pieruzzi, C Crema, A Malliani, and S Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. In *Computers in Cardiology 1998*, pages 73–76. IEEE, 1998.

[53] John Aach and George M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

[54] Yunyue Zhu and Dennis Shasha. Warping indexes with envelope transforms for query by humming. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 181–192. ACM, 2003.

[55] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-015157-2.

[56] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. *Third Workshop on Mining Temporal and Sequential Data*, pages 22–25, 2004. URL http://www.cs.ucr.edu/~eamonn/DTW_myths.pdf.

[57] Chotirat Ann Ratanamahatana and Eamonn Keogh. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005. ISSN 14451336. DOI 10.1007/s10115-004-0154-9.

[58] Derick A. Johnson and Mohan M. Trivedi. Driving style recognition using a smartphone as a sensor platform. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 1609–1615, 2011. DOI 10.1109/ITSC.2011.6083078.

[59] N Pandria and D Kugiumtzis. Testing the correlation of time series using dynamic time warping.

[60] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79, 1999.

[61] J Dai, J Teng, and X Bai. Mobile phone based drunk driving detection. *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, page 1–8, 2010. DOI 10.4108/ICST.PERVASIVEHEALTH2010.8901.

[62] Artis Mednis, Girts Strazdins, and Georgijs Kanonirs. Towards Vehicular Sensor Networks with Android Smartphones for Road Surface Monitoring. *2Nd International Workshop on Networks of Cooperating Objects*, pages 1–4, 2011. URL http://strazdins.lv/papers/strazdins2011androidpotholes.pdf.

[63] Viengnam Douangphachanh and Hiroyuki Oneyama. A study on the use of smartphones under realistic settings to estimate road roughness condition. *Proceedings of the Eastern Asia Society for Transportation Studies*, 9(2007):14, 2013. ISSN 1881-1124. DOI 10.1186/1687-1499-2014-114.

[64] James Theiler, Stephen Eubank, André Longtin, Bryan Galdrikian, and J. Doyne Farmer. Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1):77 – 94, 1992. ISSN 0167-2789. DOI 10.1016/0167-2789(92)90102-S.

[65] Thomas Schreiber and Andreas Schmitz. Improved surrogate data for nonlinearity tests. *Phys. Rev. Lett.*, 77:635–638, Jul 1996. DOI 10.1103/PhysRevLett.77.635.

[66] Martin Fowler. GUI Architectures, Jul 2006. URL https://www.martinfowler.com/eaaDev/uiArchs.html.

[67] J.D. Evans. *Straightforward Statistics for the Behavioral Sciences*. Brooks/Cole Publishing Company, 1996. ISBN 9780534231002.

A

MOBILE DEVICES SENSORS

Mobile devices, a category where smartphones and tablets are included, are now commonplace to nearly half of world's population [18]. They offer a great many of new capabilities and use cases, some of them being provided by new – or much improved – sensors. The perceived increase in the number of sensors per mobile device is pretty obvious to an attentive observer, but it was not possible to find any studies to confirm this intuition.

To investigate which sensors were more prevalent on mobile devices and try to spot any trends in the recent years, it was decided to search data on this subject on GSM Arena.[1] This is a website specialized on gathering and listing mobile devices specifications, ranked by Alexa[2] as of the writing of this document between the Top 300 and Top 400 websites.

After some examination, an automated script was written to fetch data from every mobile device listed on the website. As a result, a database was created with information about close to eight thousand mobile devices, more than twenty sensors, and upwards of twenty-five thousand relations between those mobile devices and sensors.

With this kind of dataset, it was possible to extract some insights which will be following detailed in a series of charts.

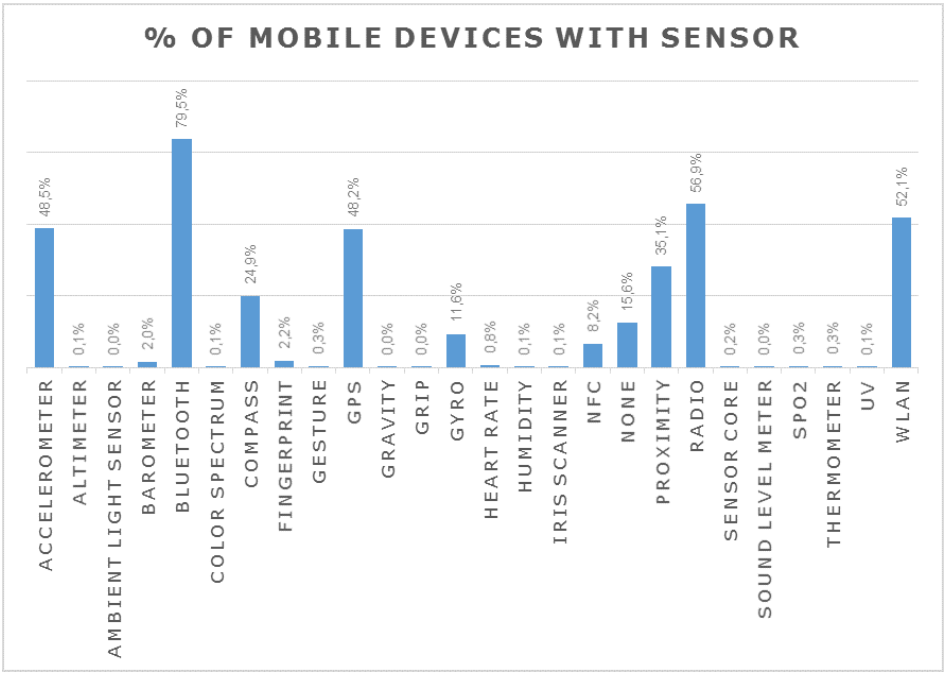
About 16 % of the mobile devices are reported to have no sensors at all; but this is changing at a very rapid pace, with that number dropping to less than half a percent if only devices still available for purchase are taken into consideration.

Present on 79 % of mobile devices, Bluetooth is the most widely used sensor by a very large margin – radio comes second place, with almost 57 %. Time has made Bluetooth virtually ubiquitous, being part of about 97 % of the mobile devices still available -– WLAN overcame radio as the second placed sensor, with 82 %.

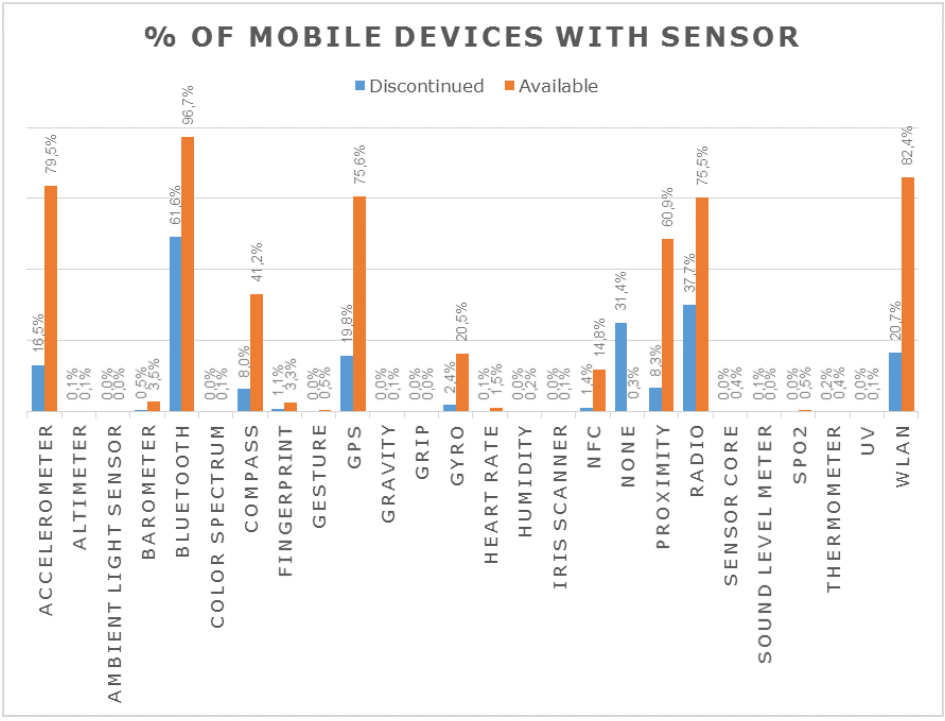
For the group of mobile devices still available, the total of sensors with a presence greater than 50 % doubled, rising from three to six: accelerometer, GPS, and proximity joined the likes of Bluetooth, radio, and WLAN.

Accelerometer (31 p.p.), GPS (27 p.p.), proximity (26 p.p.), and WLAN (30 p.p.) are the sensors with differences between percentages on both tables greater than 25 p.p.

1 http://www.gsmarena.com/
2 http://www.alexa.com/siteinfo/www.gsmarena.com



(a)



(b)

Figure 15: (a) Percentage of mobile devices with sensors. (b) Shows the same data as (a) but splitting mobile devices by availability for purchase at the time of analysis.

B

DATA ACQUISITION REQUIREMENTS

This section describes both the functional and non-functional requirements identified for the Android application responsible for collecting data from the Android device, storing it and making it available to analysis in a number of methods.

1. Description – Researcher with Android device reads GPS sensor and values are displayed in a view.
   - Rationale – The researcher needs to be able to read GPS data in real time before starting a recording session so they are able to inspect the output and make sure it is in accordance with the expected values.
   - Acceptance criteria – Researcher sees latitude, longitude, and current speed, in the app, updated at least every second.
2. Description – Researcher with Android device reads accelerometer sensor for raw data and its values are displayed in a view.
   - Rationale – The researcher needs to be able to read raw accelerometer data in real time before starting a recording session so they are able to inspect the output and make sure it is in accordance with the expected values.
   - Acceptance criteria – Researcher sees raw acceleration values for x, y, and z axles in the app, updated at least every 0.25 seconds.
3. Description – Researcher with Android device accesses linear accelerometer software-based sensor and its values are displayed in a view. This sensor provides data according to the following relationship:

   linear acceleration $=$ acceleration $-$ acceleration due to gravity

   - Rationale – The researcher needs to be able to read linear accelerometer data in real time before starting a recording session so they are able to inspect the output and make sure it is in accordance with the expected values.
   - Acceptance criteria – Researcher sees linear acceleration values for x, y, and z axles int the app, updated at least every 0.25 seconds.

4. Description – Researcher with Android device obtains a list of all available sensors in the device in a view.
    - Rationale – The researcher needs to inspect the available sensors in any given Android device in order to ensure it meets the criteria to perform a recording session.
    - Acceptance criteria – Researcher sees a list of all available sensors in a given Android device.
5. Description – Researcher with Android device obtains the name, description, delay values, and vendor for each of the available sensors in a view.
    - Rationale – The researcher needs to inspect details associated with any of the available sensors from a given Android device in order to ensure it meets the criteria to perform a recording session.
    - Acceptance criteria – Researcher sees details of any of the sensors available on a given Android device.
6. Description – Researcher with Android device starts and stops reading and recording sensors data.
    - Rationale – The researcher needs to control when the Android device starts and stops reading and recording sensors data to avoid an excessive amount of data and wasted resources.
    - Acceptance criteria – Researcher sees a button with the indication to start reading and recording sensors data, touches that button and sees a visual indication on the Android device stating the action was performed with success; then, the researcher sees a button with the indication to stop reading and recording sensors data, touches that button and sees a visual indication on the Android device stating the action was performed with success.
7. Description – Researcher with Android device records sensors data in a session-based approach. A session is a time bounded recording of sensors data with a timestamp for its beginning and end.
    - Rationale – The researcher need to know when it started and finished recording sensors data in order to aggregate data in time boxes representing a recording session.
    - Acceptance criteria – After starting recording sensors data, researcher sees a new session entry in a list of session stating the time data started to be recorded; after finishing recording sensors data, researcher sees the current session data updated to show both the time data started and finished being recorded.

8. Description – Researcher with Android device sends sensed data in near real-time to a backend service.
    - Rationale – The researcher need sensed data to be available in a backend service in order to further analyze it.
    - Acceptance criteria – The researcher accesses a backend service an it able to access data being sent to it by an Android device.
9. Description – Researcher with Android device automatically stores sensed data on the Android device.
    - Rationale – The researcher need to stored the sensed data on the Android device in order to have a backup for the data and later be able to send it to a backend service or export it to a file.
    - Acceptance criteria – Sensed data is recorded in a database on the Android device.
10. Description – Researcher with Android device has the option to choose connectivity for sending data (LTE and/or WiFi).
    - Rationale – The researcher need to choose between connectivity option for sending the sensed data to a backend service to allow in order to prevent exhausting connectivity resources when sending a large amount of data.
    - Acceptance criteria – Researcher sees the option to choose between connectivity options, selects one of them, sends data to the backend service and is able to tell if the chosen option was used.
11. Description – Researcher with Android device configures which sensors to record and the corresponding sampling rate.
    - Rationale – The researcher needs to control when the Android which sensors to record data and their corresponding sampling rate to avoid an excessive amount of data and wasted resources.
    - Acceptance criteria – Researcher configures which sensors to record and the corresponding sampling rate, starts recording data, then finishes recording data, and then sees a log of sensed data in the backend service corresponding to the configurations set by him.
12. Description – Researcher with Android device accesses a log of the activities performed by him in the app (e.g., configurations, exceptions, start/stop recording).
    - Rationale – The researcher need to keep a record of the activities performed by him in the app to help they solve eventual problems by revisiting their steps.

- Acceptance criteria – Researcher performs predetermined actions in the app, then navigates to the activities' log and seed all of the actions in the reverse order they were performed.

13. Description – Researcher with Android device calibrates the Android device sensors before starting to record session data.
    - Rationale – The researcher needs to take into account the subtle differences each of the Android devices' sensors have when recording data.
    - Acceptance criteria – Researcher performs an action that triggers a calibration and then sees a visual confirmation that it happened with success.

C

TIMEWARPER IMPLEMENTATION

This section presents the source code for TimeWaper, a Java application built to automate the data analysis performed during the dissertation. A description of its features can be found in Section 3.8.3, so this is mostly about the implementation of the tool.

The relevant classes are presented bellow along with a brief description of their responsibilities. A few classes – like `FirstRun` or `ThirdRun` – are omitted to avoid this section being even longer, but their interface and a base or exemplary class is presented.

The `import` statements were condensed to enhance the readability of the source code. Such statements should be easy to recreate with the help of an integrated development environment (IDE). Also to improve readability, the number of indenting spaces was cut to half – two spaces per indenting level. File paths were redacted for privacy reasons.

```
1  package main;
   // import ...
   public class TimeWarper {

5    public static void main(String[] args) throws IOException {
       List<Run> runs = new ArrayList<>();
       runs.add(new FirstRun());
       runs.add(new SecondRun());
       runs.add(new ThirdRun());
10     runs.add(new FourthRun());
       runs.add(new FifthRun());

       for (Run run : runs) {
         List<Session> sessions = run.getSessions();
15       for (Session session : sessions) {
           String phoneAFile = session.getSmartphoneA();
           TimeSeries seriesA = TimeSeriesCsvParser.parseFile(phoneAFile);
           String phoneBFile = session.getSmartphoneB();
           TimeSeries seriesB = TimeSeriesCsvParser.parseFile(phoneBFile);
20         String logFile = "REDACTED.csv";
           ResultsLogger logger = new ResultsLogger(logFile);
```

```
          Runnable tester = new SignificanceTester(seriesA, seriesB, logger);
          tester.run();
        }
      }
    }
}
```

Listing 1: `TimeWarper` class encloses the `main()` method, where experiments are set up and performed

```
package data;
// import ...
public interface Run {

  List<Session> getSessions();
}
```

Listing 2: `Run` interface declares a method to fetch the sessions associated with it

```
package data;
// import ...
public class SecondRun implements Run {

  private static final String SESSION_NAME_PREFIX = "run-2-session-";
  private static final String SESSION_PATH_PREFIX = "REDACTED";
  private static final String SMARTPHONE_A = "dPAob9l_wKA";
  private static final String SMARTPHONE_B = "dZLK-SbACzk";

  @Override
  public List<Session> getSessions() {
    List<Session> sessions = new ArrayList<>();

    for (int sessionNumber = 1; sessionNumber <= 5; sessionNumber++) {
      sessions.add(new SessionBase(
          SESSION_NAME_PREFIX + sessionNumber,
          SESSION_PATH_PREFIX + sessionNumber + "\\",
          sessionNumber,
          SMARTPHONE_A,
          SMARTPHONE_B));
    }

    return sessions;
  }
}
```

Listing 3: `SecondRun` class implements the `Run` interface, returns all the successful recording sessions associated with it

```
package data;

public interface Session {

  String getSessionName();

  String getSmartphoneA();

  String getSmartphoneB();
}
```

Listing 4: `Session` interface declares methods to fetch information about it

```
package data;

public class SessionBase implements Session {

  private final String mSessionName;
  private final String mPath;;
  private final int mSessionNumber;
  private final String mSmartphoneA;
  private final String mSmartphoneB;

  SessionBase(
      String sessionName,
      String path,
      int sessionNumber,
      String smartphoneA,
      String smartphoneB) {
    mSessionName = sessionName;
    mPath = path;
    mSessionNumber = sessionNumber;
    mSmartphoneA = smartphoneA;
    mSmartphoneB = smartphoneB;
  }

  @Override
  public String getSessionName() {
    return mSessionName;
  }

  @Override
  public String getSmartphoneA() {
```

```
    return mPath + "session-" + mSessionNumber + "-" + mSmartphoneA + ".csv";
  }

  @Override
  public String getSmartphoneB() {
    return mPath + "session-" + mSessionNumber + "-" + mSmartphoneB + ".csv";
  }
}
```

Listing 5: `SessionBase` class defines a base implementation of the `Session` interface

```
package main;
// import ...
public class SignificanceTester implements Runnable {

  private static final int NUMBER_OF_SURROGATES = 100;
  // The seed is fixed to allow for reproducible results
  private final AtomicLong mSeed = new AtomicLong(123456789);
  private final TimeSeries mSeriesA;
  private final TimeSeries mSeriesB;
  private final List<TimeSeries> mSurrogatesA;
  private final List<TimeSeries> mSurrogatesB;
  private final ResultsLogger mLogger;

  public SignificanceTester(
      TimeSeries seriesA, TimeSeries seriesB, ResultsLogger logger) {
    mSeriesA = seriesA;
    mSeriesB = seriesB;
    mSurrogatesA = buildTimeSeriesSurrogates(mSeriesA, mSeed);
    mSurrogatesB = buildTimeSeriesSurrogates(mSeriesB, mSeed);
    mLogger = logger;
  }

  @Override
  public void run() {
    final int processors = Runtime.getRuntime().availableProcessors();
    System.out.print("\nAvailable processors:\t" + String.valueOf(processors));

    System.out.print("\nComputing correlations\t");
    List<BigDecimal> correlations = new ArrayList<>(NUMBER_OF_SURROGATES + 1);
    new Correlator(mSeriesA, mSeriesB, correlations).run();
    final BigDecimal correlationValueOfOriginalPair = correlations.get(0);

    final int nThreads = ((processors / 2) > 0) ? (processors / 2) : 1;
    final ExecutorService executor = Executors.newFixedThreadPool(nThreads);
    for (int i = 0; i < NUMBER_OF_SURROGATES; i++) {
      Runnable task = new Correlator(
```

```
          mSurrogatesA.get(i), mSurrogatesB.get(i), correlations);
      executor.submit(task);
    }
    Util.shutdownAndAwaitTermination(executor);

    Collections.sort(correlations);
    final int originalPairIndex =
        correlations.indexOf(correlationValueOfOriginalPair);

    mLogger.log(correlations);
    System.out.println("Correlation value of original pair: "
        + String.valueOf(correlationValueOfOriginalPair.doubleValue()));
    System.out.println("Index of original pair: "
        + String.valueOf(originalPairIndex + 1) + '/'
        + String.valueOf(correlations.size()));
  }

  private static final List<TimeSeries> buildTimeSeriesSurrogates(
      TimeSeries originalSeries, AtomicLong seed) {
    final List<TimeSeries> surrogates = new ArrayList<>(NUMBER_OF_SURROGATES);

    System.out.print("\nBuilding surrogates\t");
    for (int i = 0; i < NUMBER_OF_SURROGATES; i++) {
      final Random random = new Random(seed.getAndIncrement());
      surrogates.add(new TimeSeriesShufflingWrapper(originalSeries, random));
      System.out.print('.');
    }

    return surrogates;
  }

  private static class Correlator implements Runnable {

    private final TimeSeries mSeriesI;
    private final TimeSeries mSeriesJ;
    private final List<BigDecimal> mCorrelations;

    Correlator(
        TimeSeries seriesI, TimeSeries seriesJ, List<BigDecimal> correlations) {
      mSeriesI = seriesI;
      mSeriesJ = seriesJ;
      mCorrelations = correlations;
    }

    @Override
    public void run() {
      final Correlation correlation = WarpedCorrelation.build(mSeriesI, mSeriesJ);
```

```
        mCorrelations.add(correlation.correlate());
        System.out.print('.');
      }
    }

    static class ResultsLogger {

      private final String mFileName;

      ResultsLogger(String fileName) {
        mFileName = fileName;
      }

      public void log(List<BigDecimal> results) {
        Path file = Paths.get(mFileName);
        List<String> resultsAsText = results
            .stream()
            .map(result -> String.valueOf(result.doubleValue()))
            .collect(Collectors.toList());

        try {
          Files.write(file, resultsAsText, Charset.forName("UTF-8"));
        } catch (IOException e) {
          System.out.println("\nCouldn't log results to file '" + mFileName
              + "':\n" + results.toString());
        }
        System.out.println("Results logged to: ".concat(mFileName));
      }
    }
}
```

Listing 6: `SignificanceTester` class build surrogates when given two `TimeSeries` objects. Those time series are then processed with a DTW algorithm and their correlation is computed and logged. This file includes two static nested classes, `Correlator` and `ResultsLogger`, used to keep the code decoupled and allowing for dependency injection. The `Correlator` class has the responsibility of computing the cross-correlation by using the `WarpedCorrelation` class, which extends the `Correlation` class. This class also performs the statistical validation of the results, exporting it to a file using the `ResultsLogger` class

```
package main;
// import ...
public class Correlation {

  protected static final int CORRELATING_VALUE_INDEX = 0;
  private static final int SQUARED = 2;
```

```
  private final TimeSeries mSeriesA;
  private final TimeSeries mSeriesB;
  private final BigDecimal mSeriesAAvg;
  private final BigDecimal mSeriesBAvg;

  public Correlation(TimeSeries timeSeriesA, TimeSeries timeSeriesB) {
    if (timeSeriesA.size() != timeSeriesB.size()) {
      throw new RuntimeException("Time series have different sizes");
    }

    mSeriesA = timeSeriesA;
    mSeriesB = timeSeriesB;
    mSeriesAAvg = averageOf(mSeriesA);
    mSeriesBAvg = averageOf(mSeriesB);
  }

  public final BigDecimal correlate() {
    BigDecimal productOfDiffToMeans = BigDecimal.ZERO;
    BigDecimal diffToMeanSquaredA = BigDecimal.ZERO;
    BigDecimal diffToMeanSquaredB = BigDecimal.ZERO;

    final int numberOfValues = mSeriesA.size();
    for (int i = 0; i < numberOfValues; i++) {
      productOfDiffToMeans = productOfDiffToMeans.add(
          productOfDiffToMeansAtPosition(
          mSeriesA, mSeriesB, i, mSeriesAAvg, mSeriesBAvg));
      diffToMeanSquaredA = diffToMeanSquaredA.add(
          diffToMeanSquaredAtPosition(mSeriesA, i, mSeriesAAvg));
      diffToMeanSquaredB = diffToMeanSquaredB.add(
          diffToMeanSquaredAtPosition(mSeriesB, i, mSeriesBAvg));
    }

    final BigDecimal productOfDiffToMeansSquared =
        diffToMeanSquaredA.multiply(diffToMeanSquaredB);
    final double divisor = Math.sqrt(productOfDiffToMeansSquared.doubleValue());

    return productOfDiffToMeans.divide(
        BigDecimal.valueOf(divisor), MathContext.DECIMAL128);
  }

  private static final BigDecimal averageOf(TimeSeries series) {
    final int seriesSize = series.size();
    BigDecimal sum = BigDecimal.ZERO;

    for (int i = 0; i < seriesSize; i++) {
      final double measurement = series.getMeasurement(i, CORRELATING_VALUE_INDEX);
      final BigDecimal seriesValue = BigDecimal.valueOf(measurement);
```

```
55          sum = sum.add(seriesValue);
          }

          return sum.divide(BigDecimal.valueOf(seriesSize), MathContext.DECIMAL128);
        }
60
        private static final BigDecimal productOfDiffToMeansAtPosition(
            TimeSeries seriesA,
            TimeSeries seriesB,
            int position,
65          BigDecimal seriesAAvg,
            BigDecimal seriesBAvg) {
          final BigDecimal diffToMeanA = BigDecimal
              .valueOf(seriesA.getMeasurement(position, CORRELATING_VALUE_INDEX))
              .subtract(seriesAAvg);
70        final BigDecimal diffToMeanB = BigDecimal
              .valueOf(seriesB.getMeasurement(position, CORRELATING_VALUE_INDEX))
              .subtract(seriesBAvg);

          return diffToMeanA.multiply(diffToMeanB);
75      }


        private static final BigDecimal diffToMeanSquaredAtPosition(
            TimeSeries series, int position, BigDecimal mean) {
80        return BigDecimal
              .valueOf(series.getMeasurement(position, CORRELATING_VALUE_INDEX))
              .subtract(mean)
              .pow(SQUARED);
        }
85  }
```

Listing 7: `Correlation` class computes the correlation between two time series with the same size

```
1   package main;
    // import ...
    public class WarpedCorrelation extends Correlation {

5     private static final int DEFAULT_SEARCH_RADIUS = 1000;

      private WarpedCorrelation(
          TimeSeries seriesA, TimeSeries seriesB, WarpPath path) {
        super(Warper.warpTimeSeries(seriesA, path, Warper.POSITION_IN_PATH.COLUMN),
10          Warper.warpTimeSeries(seriesB, path, Warper.POSITION_IN_PATH.ROW));
      }

      public static final WarpedCorrelation build(
```

```
      TimeSeries seriesA, TimeSeries seriesB) {
    return build(seriesA, seriesB, Distances.EUCLIDEAN_DISTANCE);
  }

  public static final WarpedCorrelation build(
      TimeSeries seriesA, TimeSeries seriesB, DistanceFunction function) {
    TimeWarpInfo info = FastDTW.compare(
        seriesA, seriesB, DEFAULT_SEARCH_RADIUS, function);
    return new WarpedCorrelation(seriesA, seriesB, info.getPath());
  }

  static class Warper {

    static enum POSITION_IN_PATH {
      COLUMN, ROW
    };

    static TimeSeries warpTimeSeries(
        TimeSeries series, WarpPath path, POSITION_IN_PATH mode) {
      final int pathSize = path.size();
      final TimeSeriesBase.Builder seriesBuilder = new TimeSeriesBase.Builder();

      for (int i = 0; i < pathSize; i++) {
        final int warpedIndex = getWarpedIndex(path, i, mode);
        seriesBuilder.add(
            i, series.getMeasurement(warpedIndex, CORRELATING_VALUE_INDEX));
      }

      return seriesBuilder.build();
    }

    private static int getWarpedIndex(
        WarpPath path, int index, POSITION_IN_PATH position) {
      switch (position) {
        case COLUMN:
          return path.get(index).getCol();
        case ROW:
          return path.get(index).getRow();
        default:
          throw new RuntimeException("Position in path not recognized");
      }
    }
  }
}
```

Listing 8: `WarpedCorrelation` extends `Correlation` and uses DTW to warp the two time series before computing their correlation

```java
package main;
// import ...
public final class TimeSeriesCsvParser {

  private static final String LINE_SPLITTER = ";";
  private static final int COLUMN_ACCELEROMETER_Y_AXIS_IDX = 2;
  private static final int COLUMN_MEASURED_AT_IDX = 10;
  private static final int COLUMN_EVENT_INDEX = 13;
  private static final String EVENT_START_SESSION = " Start session";
  private static final String EVENT_END_SESSION = " End session";

  private TimeSeriesCsvParser() {
  }

  public static final TimeSeries parseFile(String pathname) throws IOException {
    return parseFile(new File(pathname));
  }

  public static final TimeSeries parseFile(File file) throws IOException {
    final TimeSeriesBase.Builder seriesBuilder = TimeSeriesBase.builder();
    final BufferedReader reader = new BufferedReader(new FileReader(file));

    reader.readLine(); // skip the header
    String line = reader.readLine();
    AtomicBoolean reachedStartOfSession = new AtomicBoolean(false);
    AtomicBoolean reachedEndOfSession = new AtomicBoolean(false);
    long linesAdded = 0L;

    while (line != null) {
      String[] columns = line.split(LINE_SPLITTER);
      line = reader.readLine();

      final String event = columns[COLUMN_EVENT_INDEX];
      if (shouldSkipLine(event, reachedStartOfSession, reachedEndOfSession)) {
        continue;
      }

      final double measuredAt = Double
          .valueOf(columns[COLUMN_MEASURED_AT_IDX].replace(',', '.'));
      final double accelerometerYAxis = Double
          .valueOf(columns[COLUMN_ACCELEROMETER_Y_AXIS_IDX].replace(',', '.'));
      seriesBuilder.add(measuredAt, accelerometerYAxis);
      linesAdded++;
    }

    final String linesAddedMessage = "\nLines added in this session: "
        .concat(String.valueOf(linesAdded));
```

```
      System.out.print(linesAddedMessage);
      reader.close();
      return seriesBuilder.build();
    }

    // AtomicBooleans reachedStartOfSession and reachedEndOfSession are mutated
    // inside this method
    // Here be dragons
    private static final boolean shouldSkipLine(
        final String event,
        AtomicBoolean reachedStartOfSession,
        AtomicBoolean reachedEndOfSession) {
      if (!reachedStartOfSession.get()) {
        reachedStartOfSession.set(EVENT_START_SESSION.equals(event));
      }
      if (!reachedEndOfSession.get()) {
        reachedEndOfSession.set(EVENT_END_SESSION.equals(event));
      }

      final boolean withinSessionBounds =
          reachedStartOfSession.get() && (!reachedEndOfSession.get());
      return !withinSessionBounds;
    }
  }
```

Listing 9: `TimeSeriesCsvParser`, a utility-like class to read a file with session data and build a `TimeSeries` with the vertical acceleration data

```
package main;
// import ...
public class Util {

  private static final int TIMEOUT_SIZE = 600
  private static final TimeUnit TIMEOUT_UNIT = TimeUnit.MINUTES;

  private Util() {
  }

  public static void shutdownAndAwaitTermination(ExecutorService executor) {
    executor.shutdown();
    try {
      if (!executor.awaitTermination(TIMEOUT_SIZE, TIMEOUT_UNIT)) {
        executor.shutdownNow();
        System.out.println("\nExecutor service terminating early!");
        if (!executor.awaitTermination(TIMEOUT_SIZE, TIMEOUT_UNIT)) {
          System.out.println("\nExecutor service did not terminate!");
        }
```

```
      } else {
        System.out.println("\nExecutor service terminated successfully");
      }
    } catch (InterruptedException e) {
      executor.shutdownNow();
      Thread.currentThread().interrupt();
    }
  }
}
```

Listing 10: `Util` is a simple utility class with a single method to correctly shutdown an `ExecutorService` after it terminates its work

```
package com.fastdtw.timeseries;
// import ...
public class TimeSeriesShufflingWrapper implements TimeSeries {

  private final TimeSeries mTimeSeries;
  private final List<Integer> mShuffledIndices;

  public TimeSeriesShufflingWrapper(TimeSeries originalSeries, Random random) {
    mTimeSeries = originalSeries;
    mShuffledIndices = IntStream.range(0, originalSeries.size()).boxed()
        .collect(Collectors.toList());
    Collections.shuffle(mShuffledIndices, random);
  }

  @Override
  public int size() {
    return mTimeSeries.size();
  }

  @Override
  public int numOfDimensions() {
    return mTimeSeries.numOfDimensions();
  }

  @Override
  public double getTimeAtNthPoint(int n) {
    return mTimeSeries.getTimeAtNthPoint(getShuffledIndex(n));
  }

  @Override
  public double getMeasurement(int pointIndex, int valueIndex) {
    return mTimeSeries.getMeasurement(getShuffledIndex(pointIndex), valueIndex);
  }
```

```
35    @Override
      public double[] getMeasurementVector(int pointIndex) {
        return mTimeSeries.getMeasurementVector(getShuffledIndex(pointIndex));
      }

40    private int getShuffledIndex(int originalIndex) {
        return mShuffledIndices.get(originalIndex);
      }
  }
```

Listing 11: `TimeSeriesShufflingWrapper` class, added to the FastDTW tool to allow for a more efficient shuffle of objects implementing the `TimeSeries` interface, when working with surrogate data. Extending the `TimeSeriesBase` was another viable option, but composition was preferred over inheritance. Additionally, makes use of some functional features introduced with Java 8

D

CORRELATION COEFFICIENTS CHARTS

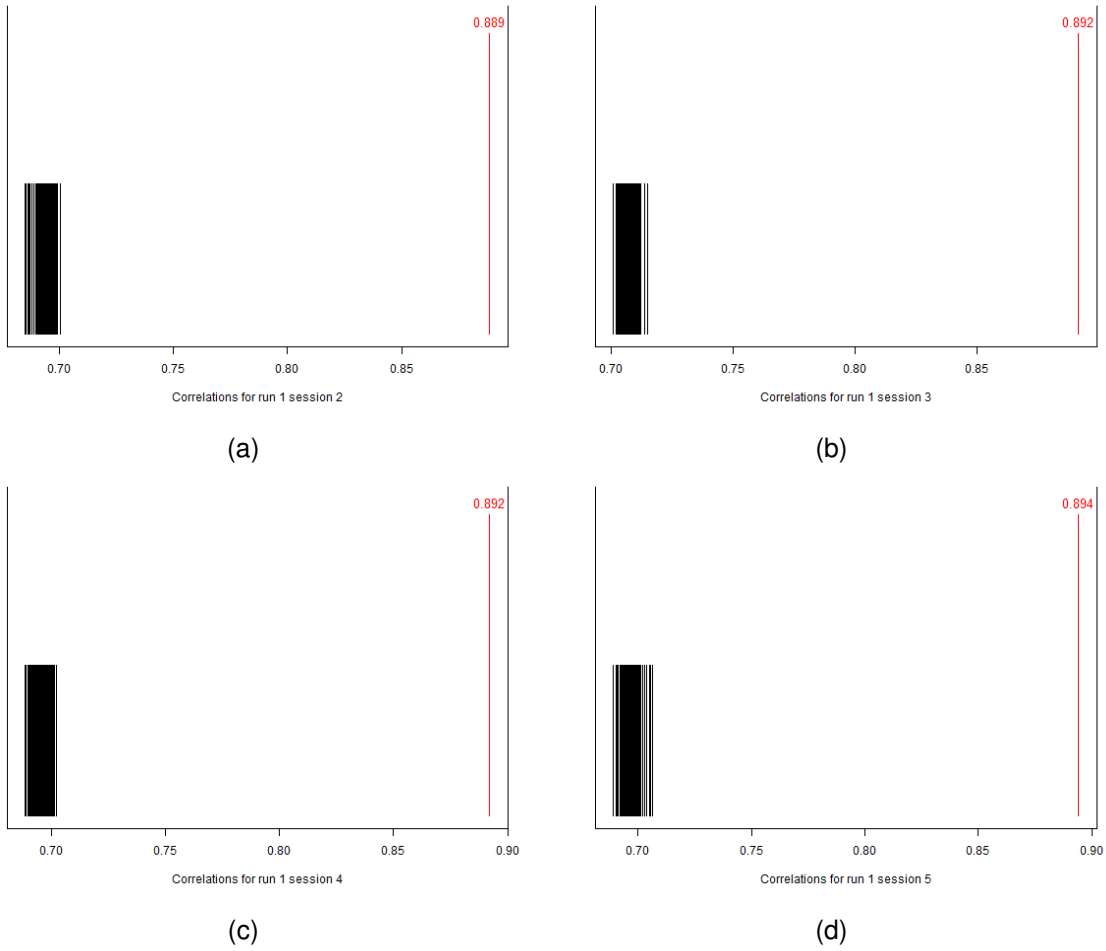


Figure 16: Correlation coefficients for the first run, including surrogate pairs and the original pair (highlighted)

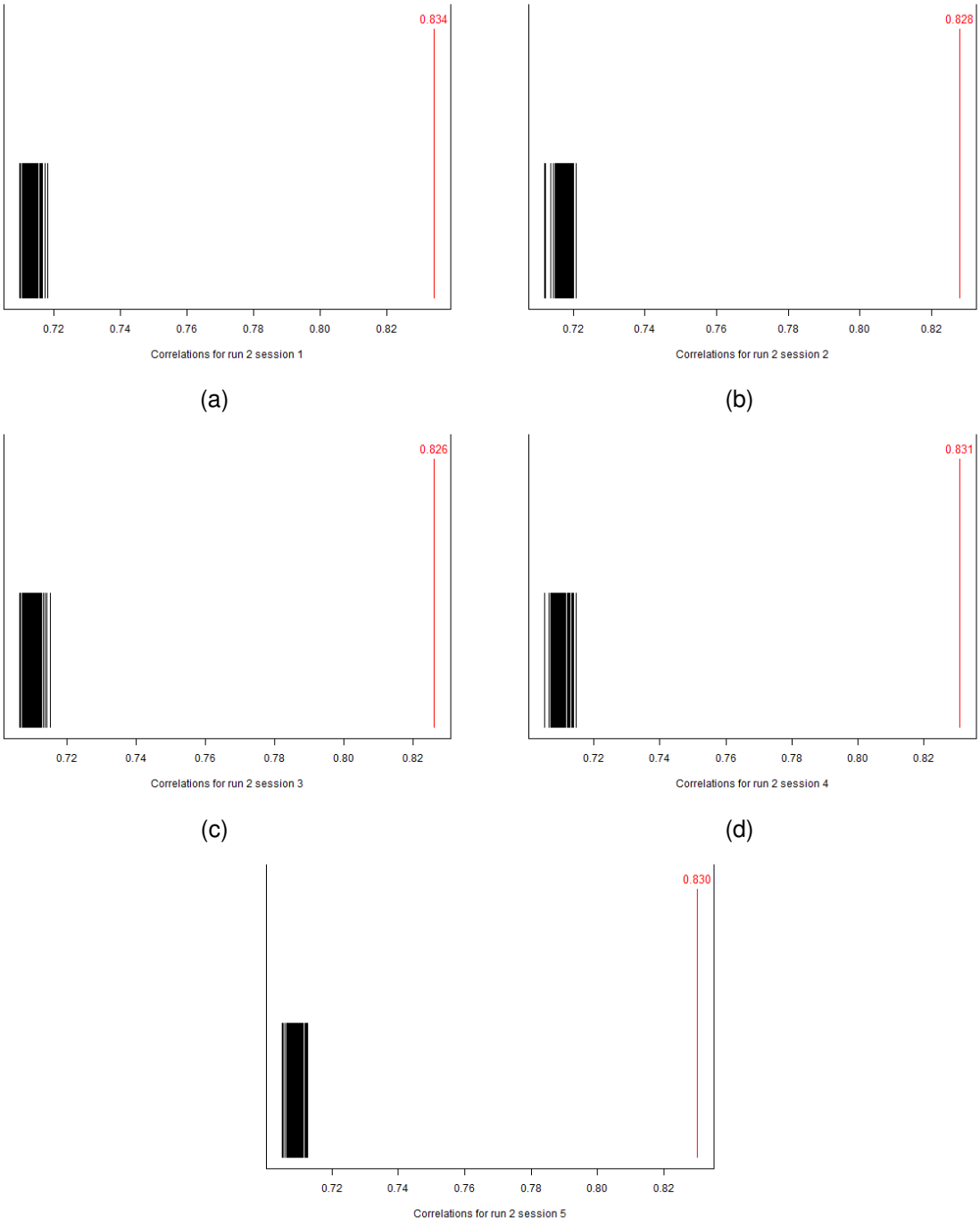


Figure 17: Correlation coefficients for the second run, including surrogate pairs and the original pair (highlighted)

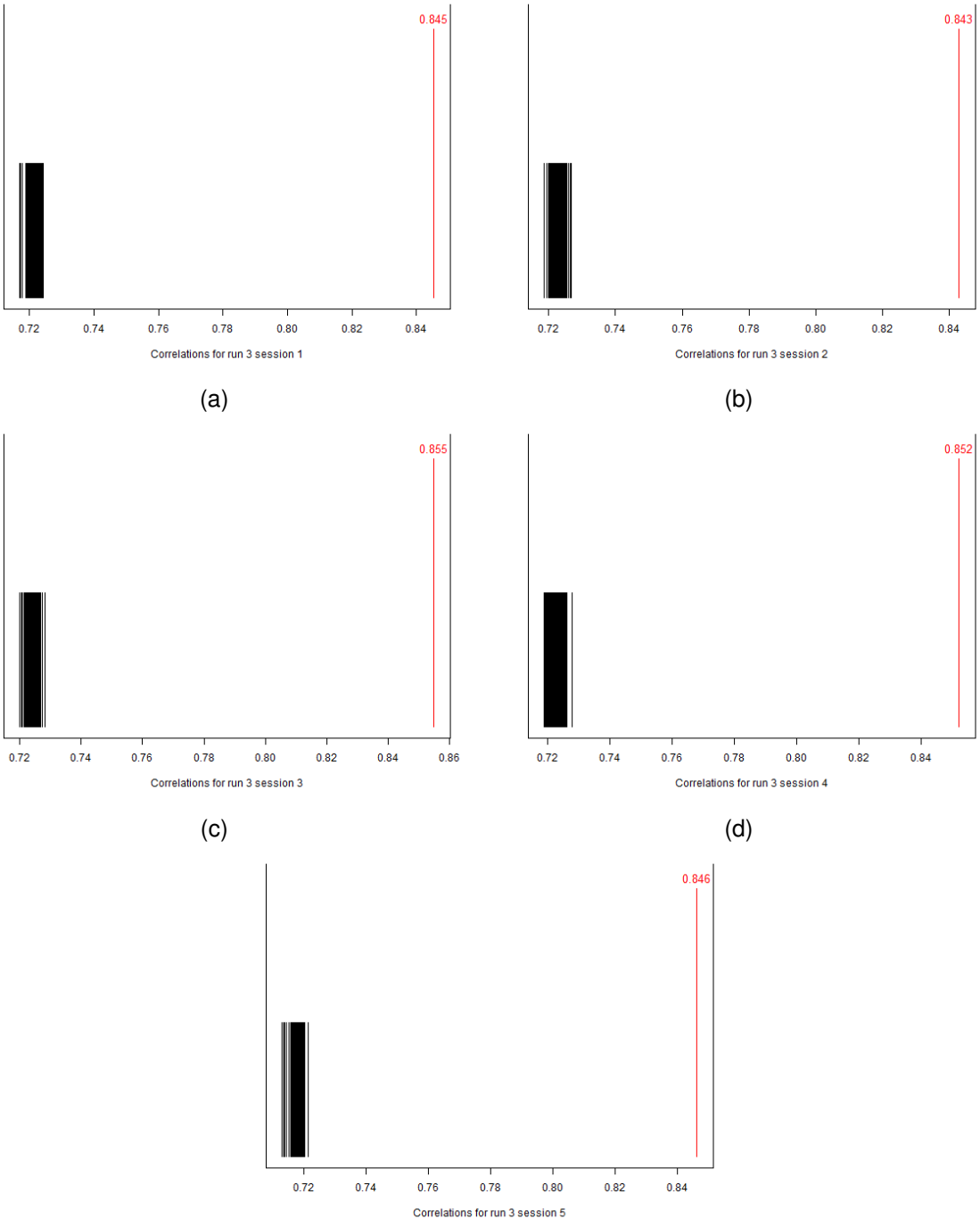


Figure 18: Correlation coefficients for the third run, including surrogate pairs and the original pair (highlighted)

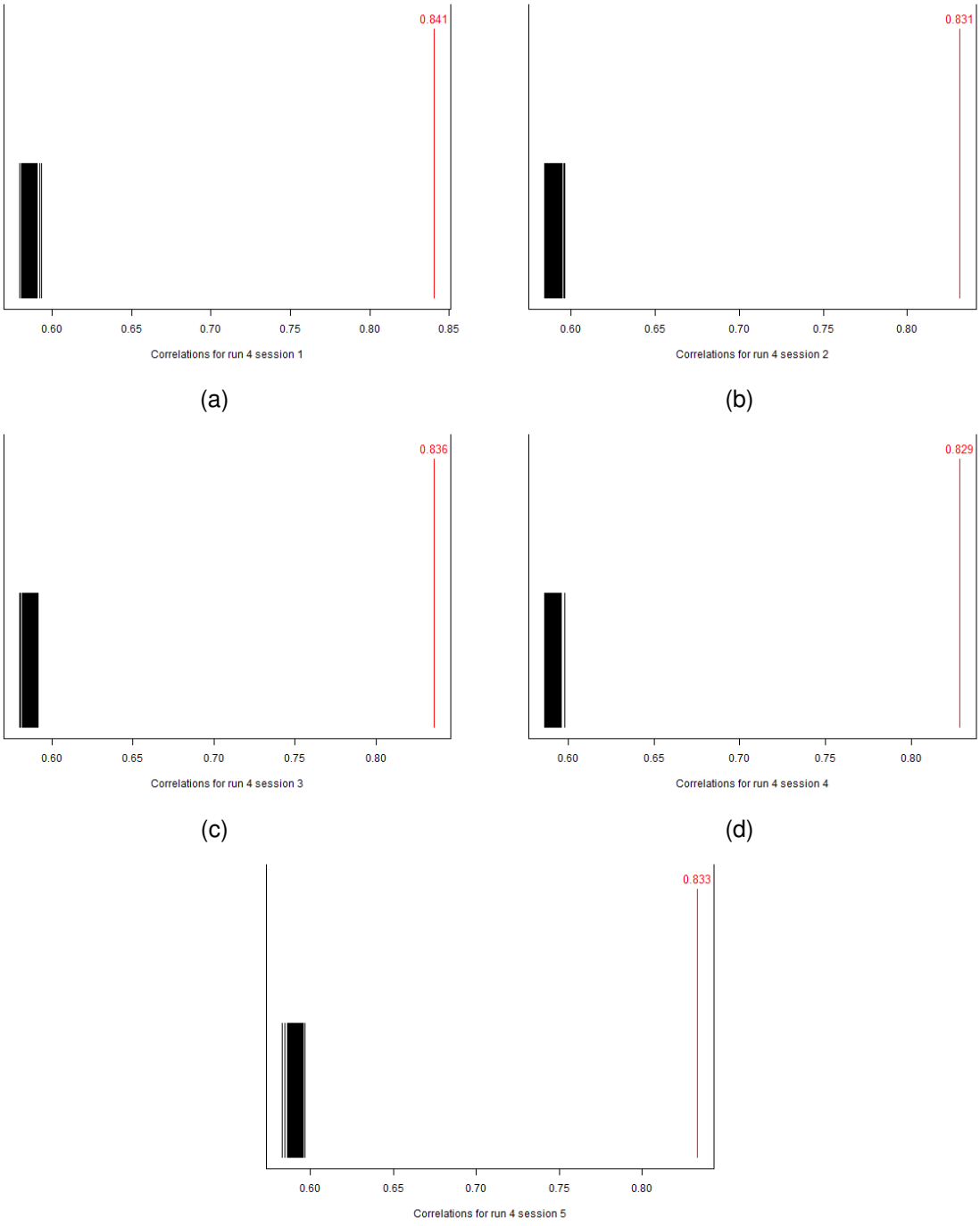


Figure 19: Correlation coefficients for the fourth run, including surrogate pairs and the original pair (highlighted)

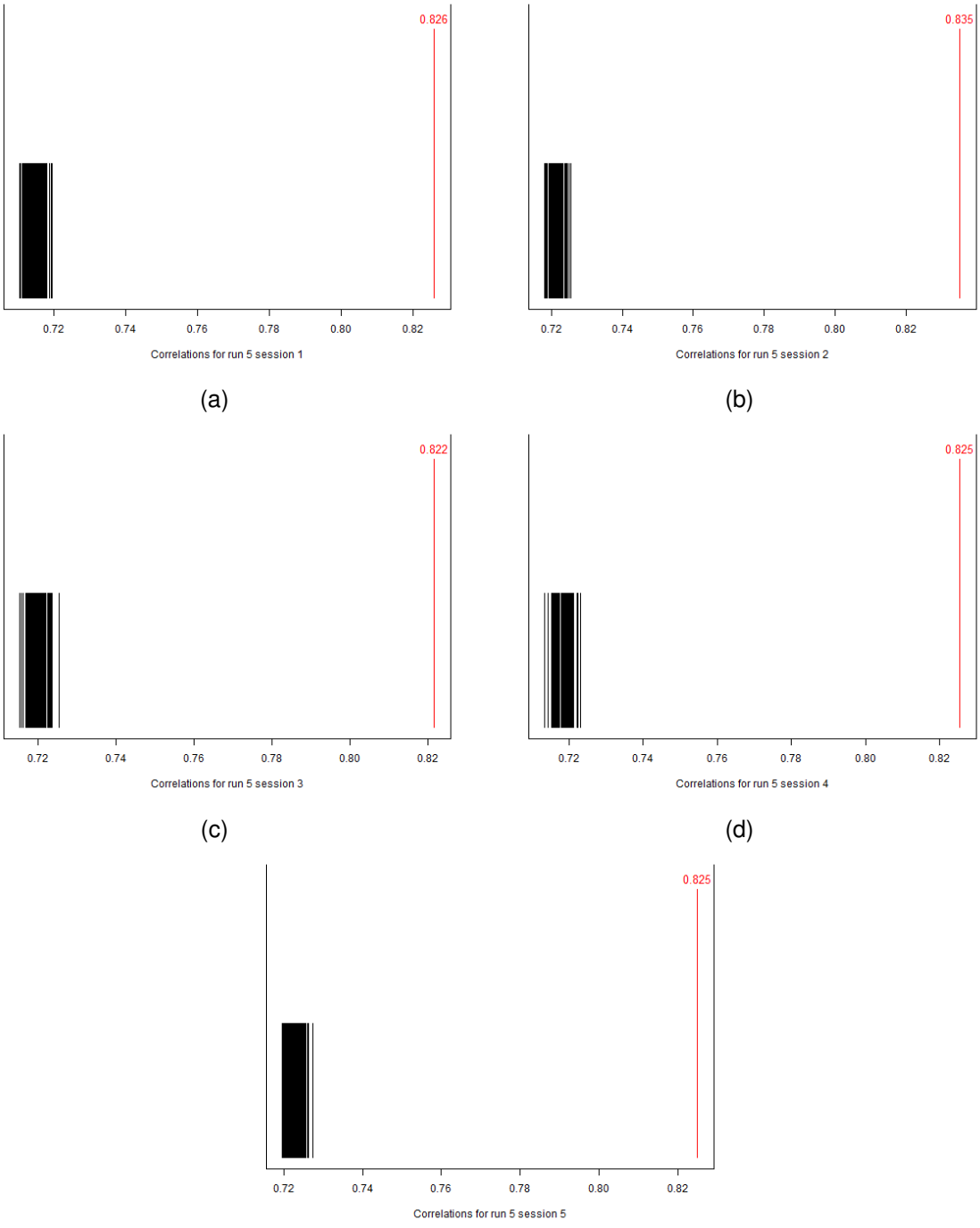


Figure 20: Correlation coefficients for the fifth run, including surrogate pairs and the original pair (highlighted)